UNIVERSITÉ DU QUÉBEC À MONTRÉAL

UNE ÉTUDE VERS LA COMPLÉTION DE SCHÉMA ET DE JEUX DE DONNÉES LIÉES À L'AIDE DE L'ANALYSE RELATIONNELLE DE CONCEPTS (ARC)

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

BILLEL SERIAI

JANVIER 2016

REMERCIEMENTS

Je tiens, tout d'abord, à remercier Monsieur Petko Valtchev, mon directeur de recherche à l'UQAM, pour sa disponibilité, son enthousiasme, ses directives et pour tous les conseils qu'il m'a prodigués tout au long de ma maitrise.

Je remercie monsieur Robert Godin qui a cru en moi en m'acceptant en tant que son étudiant avant qu'il prenne sa retraite.

Je souhaite également remercier les professeurs dont j'ai suivi les cours pendant ma maîtrise et qui ont su m'encourager et me pousser à travailler et à apprendre.

Un grand merci à Marianne Huchard pour tous les efforts qu'elle a fais afin que j'arrive à conclure ce travail.

Je tiens aussi à témoigner toute ma gratitude envers Alain Pilon qui était toujours là pour moi pendant tout mon séjour à Montréal

Je n'oublie pas mon ami Elyes Garci qui m'a épaulé et conseillé durant toute la période de mes études ici au Québec.

À mes parents, Torkia et Salah, les mots me manquent pour vous exprimer toute mon admiration et ma reconnaissance. Je vous dois tout après dieu, la vie, la réussite et le bonheur. Merci pour votre confiance et votre soutien inestimable.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AFC	Analyse Formelle de Concepts
ARC	Analyse Relationnelle de Concepts
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol and RDF Query Language
LOD	Linked Open Data

RÉSUMÉ

Les dernières années, on constate une énorme augmentation des informations publiées sur le web en tant que données liées. Cette évolution rapide crée des difficultés pour l'évaluation de la qualité et l'exploration de ces données. Dans ce mémoire, nous proposons une nouvelle approche pour surmonter ces problèmes par la mise en œuvre d'un processus de découverte de connaissances. Les sources des données structurées sur le web sont décrites par *Resources description Framework* (RDF). Nous appliquons notre approche à une ontologie déjà existante, le Dbpedia, afin de l'étudier à travers ces instances. Cette étude pourra nous révéler des incohérences ou des manques dans les données de l'ontologie. Notre approche est semi-automatique et comporte quatre phases : 1- L'extraction des données est automatique en utilisant les requêtes SPARQL. 2- La transformation des descripteurs en modèle de données, la phase étant manuelle, elle crée des descripteurs par factorisation d'attributs communs. 3- L'utilisation des résultats de l'analyse relationnelle de concepts (RCA) pour extraire un modèle de classes. 4- La comparaison des résultats avec les classes de Dbpedia. Notre approche aide à fournir un assistant pour l'extraction de données et à enrichir l'ontologie Dbpedia avec les nouvelles classes découvertes. Elle offre aussi un assistant pour la collecte des données en suggérant des valeurs pour les propriétés qui manquent.

MOTS-CLÉS : RDF, Dbpedia, Analyse Relationnelle de Concepts, Données liées,

INTRODUCTION

Les données liées accessibles (*open linked data*) sont de plus en plus nombreuses sur la toile, ouvrant la voie à de nombreuses analyses et explorations. Certaines sont créées selon des schémas prédéfinis (des ontologies par exemples), d'autres sont créées de manière plus ouverte, sans organisation de niveau schéma prédéfinie. Dbpedia est une initiative permettant de capitaliser sur des données structurées extraites de Wikipedia. Une ontologie est définie comme cadre très large, les instances d'une classe utilisant seulement une partie des données proposée par cette classe.

Dbpédia est fréquemment utilisée par des jeux de données produits par des parties tierces comme source de connaissances ontologiques, en particulier, des classes et des propriétés. Prenons par exemple la création d'un jeu de données sur les morceaux de musique qui nécessite la réutilisation des classes telles que Musicien, Morceaux, Groupe, Style, etc. déjà existantes dans Dbpedia et des propriétés telles que composéPar et jouéPar aussi existantes dans Dbpedia. Cependant, étant donné que les jeux de données peuvent par la suite être utilisés à des multiples fins, il est toujours souhaitable d'avoir un schéma bien identifié pouvant être importé dans un outil d'analyse comme dans Mehri-Dehnavi (2014) et (Paulheim et Fümkranz) (2012) ou d'intégration de données comme dans le travail de David et Scharffe (2012).

La taille de l'ontologie Dbpedia est très grande ce qui ne permet pas l'importation de la totalité des classes/propriétés. Ce n'est même pas envisageable d'importer uniquement les éléments utilisés à cause des interdépendances entre eux. Ces interdépendances peuvent être dues à des éléments non référencés. Se servir des régularités qui existent entre les instances afin de construire un schéma à partir des éléments semble être une meilleure stratégie.

D'autre part, les régularités peuvent être révélatrices de phénomènes observables uniquement sur les données. Ainsi, des cooccurrences des propriétés dans les instances peuvent, d'après leur constance, guider l'évaluation ou bien la saisie des données. En effet, même si les données sont souvent issues d'un processus de traduction automatique, en particulier sur Dbpedia, on n'est jamais à l'abri d'erreurs ni d'oublis à la source. Un certain nombre de tels erreurs ou oublis peuvent être détectés grâce à la détection de régularités ou quasi-régularités dans la distribution des propriétés dans les instances.

En somme, deux problématiques se dégagent de la manière dont interagissent données et schémas sur Dbpedia. La première consiste à observer l'usage des classes au travers des instances effectivement créées afin d'en tirer un schéma spécialisant l'ontologie et révélant son schéma tel qu'il existe en pratique. Comme seconde problématique, le schéma et d'autres observations sur l'usage sont propices à l'étude de la cohérence des données et à faire des recommandations pour les compléter ou les corriger éventuellement.

Dans cette étude, nous proposons d'aborder ces deux problématiques grâce à une méthode d'analyse de données basée sur la théorie des treillis, l'Analyse Relationnelle de Concepts (ARC). Cette méthode analyse des données composées de diverses catégories d'objets liés entre eux par des relations. De cette analyse on peut tirer d'une part des classifications des objets et d'autre part des règles d'implication. Lorsque les objets sont des descripteurs d'instances de Dbpedia, les classifications permettent d'extraire un schéma qui raffine l'ontologie initiale. De leur côté, les règles nous indiquent comment les propriétés sont utilisées et distribuées entre les classes.

Dans la suite du mémoire, nous présentons les bases techniques de l'approche en définissant les principaux concepts utilisés tout au long de cette étude : l'analyse formelle de concepts AFC, l'analyse relationnelle de concepts ARC, le *Resource Description Framework* RDF et Dbpedia (chapitre 1), un état de l'art où nous comparons notre approche à d'autres approches similaires (chapitre 2), puis la méthodologie suivie en montrant les étapes de notre approche et les résultats obtenus (chapitre 3).

CHAPITRE I
CONCEPTS DE BASE

Dans ce chapitre, nous présentons les notions de base et les définitions mathématiques relatives à l'analyse formelle de concepts (AFC) et l'analyse relationnelle de concepts (ARC). Nous décrivons aussi les données RDF. Nous représentons également la base de données Dbpedia sur laquelle nous avons appliqué notre approche. La suite du chapitre est organisée comme suit. Dans les deux premières sections, nous représentons le modèle RDF, sa syntaxe et ses domaines d'utilisation, ensuite nous détaillons Dbpedia et son ontologie. Dans les dernières sections, nous rappelons les notions de base nécessaires à la compréhension de l'analyse formelle de concepts et l'analyse relationnelle de concepts.

1.1 Logiques de descriptions (DL)

Une logique de description est un langage qui représente les connaissances en utilisant une syntaxe et une sémantique formelle. Les logiques de descriptions sont des familles de langages de représentation qui représentent les connaissances (concepts, relations, raisonnements, etc.) d'un domaine d'une manière structurée (Baader et al.) (2003).

La modélisation des connaissances d'un domaine avec les LD se réalise en deux niveaux. Un niveau terminologique *(TBox,)* décrit les connaissances générales d'un domaine comme les concepts et les rôles et un autre niveau factuel (*ABox*) représente une configuration précise comme la description des individus en les nommant et en montrant leur appartenance à des concepts ou à des rôles.

1.2 Ressources description framework (RDF)

Le *Resource Description Framework* (RDF) est un modèle de description des données qui permet de décrire les informations des ressources Web (Seligman et Roenthal, 2001). Il permet de coder, échanger et réutiliser les métadonnées structurées afin de fournir des méthodes non ambiguës exprimant la sémantique de données du Web (Miller, 1998). RDF est prévu pour être à la fois suffisamment puissant pour tout décrire, et facile à utiliser par des

logiciels. Il représente les données d'une façon formelle en utilisant des triplets sous forme de < Sujet - Prédicat - Object>. Chaque triplet est une déclaration qui décrit des ressources. Ces ressources sont identifiées dans le web par une courte chaîne de caractères appelée *identifiant uniforme de ressource* (URI). Un URI permet d'identifier une ressource d'une façon permanente, même après le déplacement ou la suppression de cette dernière.

Un triplet < Sujet – Prédicat – Objet > forme le modèle de données de base de RDF. Il se compose d'un Sujet, qui représente une ressource à décrire qui peut être une page Web, ou un objet qui n'est pas directement accessible via le Web. Le deuxième élément du triplet est le Prédicat; c'est la relation utilisée pour définir le Sujet (la ressource). Cette relation relie le premier élément du triplet au troisième élément qu'est l'Objet. L'objet représente une donnée ou une autre ressource, c'est la valeur de la propriété (Pan, 2009). Cette valeur peut être une chaîne de caractères, un nombre, etc. ou d'autres ressources qui peuvent avoir aussi leurs propres propriétés. La figure 1.1 illustre une description d'un modèle RDF.

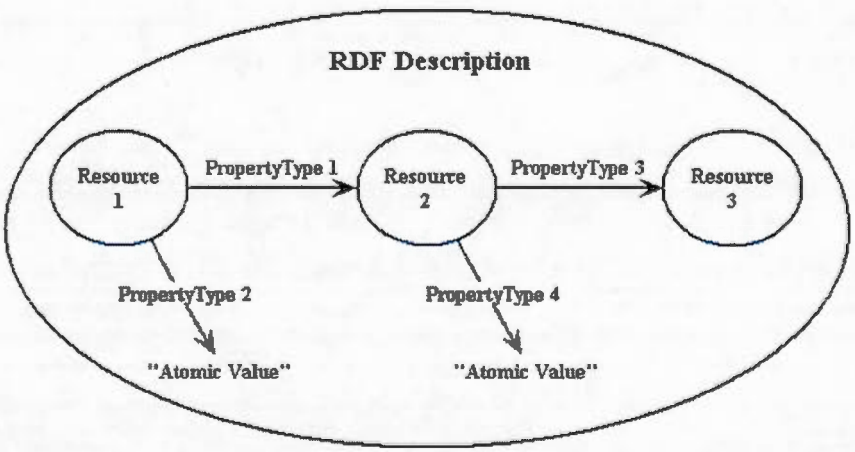

Figure 1.1 Description générique d'un modèle RDF.

La description graphique d'un modèle RDF se base sur des arcs qui représentent les propriétés. Ces arcs jouent le rôle de la relation qui lie les ressources (Sujets et Objets) représentés par des nœuds (ovales). Dans notre figure, on voit bien que deux types de valeurs sont représentés : Valeur atomique et une autre ressource.

Exemple 1 :

Dans cet exemple, on prend mon cas en tant qu'étudiant supervisé par mon directeur de recherche Monsieur Petko Valtchev.

Soit les déclarations suivantes:

" *Petko Valtchev* est un professeur qui supervise *Billel Seriai* "

" Le superviseur de *Billel Seriai* est le professeur *Petko Valtchev* "

Ces deux déclarations semblent pareilles pour un être humain car elles donnent le même sens. Contrairement à un cerveau humain, la machine les considère comme deux chaînes de caractères différentes. Dans le but d'éviter ce genre d'ambigüité, RDF fournit une façon de représenter la sémantique dans un codage lisible par la machine en utilisant un modèle de triplet < Sujet – Prédicat – Objet >. Donc les ressources (Petko Valtchev et Billel Seriai) sont représentées par des nœuds et la propriété (superviser) par un arc dirigé vers l'objet (valeur de la propriété). En appliquant ce modèle sur notre première déclaration on obtient le graphe représenté dans la figure 1.2 :

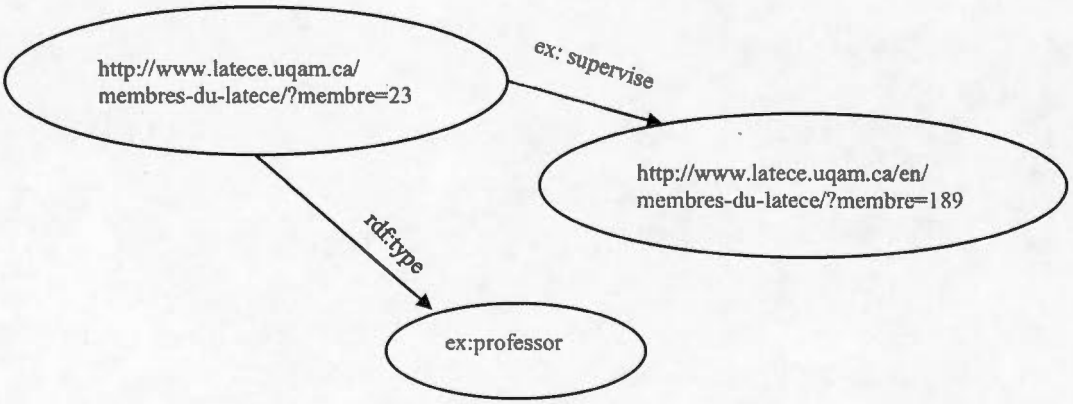

Figure 1.2 Graph RDF de l'exemple 1.

Dans la figure 1.2, les ressources (sujet et objet) sont identifiées par leurs URIs qui sont dans notre cas les URLs (*http://www.latece.uqam.ca/membres-du-latece/?membre=23*) et

(*http://www.latece.uqam.ca/en/membres-du-latece/?membre=189*) qui correspondent respectivement à la page de *Petko Valtchev* et celle de *Billel Seriai*. Ces deux ressources sont liées par la propriété (Prédicat) *Supervises*. L'espace de nom choisi dans notre exemple est "*ex*" qui représente " http://www.latece.uqam.ca/en/membres-du-latece /". L'espace de nom ***rdf*** représente le lien "http://www.w3.org/1999/02/22-rdf-syntax-ns#"

1.2.1 La syntaxe RDF/XML

RDF dispose de plusieurs syntaxes qui représentent ses modèles pour stocker les instances de ces derniers dans des fichiers lisibles pour la machine. RDF/XML est la syntaxe la plus connue car elle permet la représentation cohérente de la sémantique. Ci-dessous une représentation de notre exemple par la syntaxe RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns : rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ex =" http://www.latece.uqam.ca/en/membres-du-latece /">
<rdf:Description rdf:about=" http://www.latece.uqam.ca/membres-du-latece/?membre=23">
<rdf:type rdf:resource =" www.latece.uqam.ca/membres-du-latece /Professor"/>
<ex:supervise rdf:resource =" www.latece.uqam.ca/membres-du-latece /?membre=189"/>
</rdf:Description>
</rdf:RDF>
```

1.2.2 Domaines d'application

Le Resource Description Framework (RDF) peut être utilisé dans différents domaines tels que :

- La découverte des ressources : Les moteurs de recherche utilisent RDF pour découvrir plus facilement les ressources sur le Web.
- Le catalogage : RDF permet aux utilisateurs de mieux décrire le contenu et les relations de contenu disponibles sur le Web (Pages web, bibliothèque numérique, etc.).
- L'échange et le partage des connaissances utilisées par les agents logiciels dans le but de filtrer et fusionner des données.
- La gestion des droits de propriété intellectuelle : RDF permet aux utilisateurs de faire respecter les droits de propriété intellectuelle de sites Web (Candan et al.) (2001).

1.3 Dbpedia et son ontologie

1.3.1 Principes généraux de Dbpedia

Dbpedia[1] est un projet qui a été lancé par les universités de Berlin et de Leipzig en collaboration avec OpenLink Software. Dbpedia a pour but l'extraction de données structurées à partir de Wikipedia afin de les rendre disponibles pour l'usage public sous forme d'un modèle qui permet le traitement automatique de ces ressources. Ce modèle est connu sous le nom de RDF (Ressource description Framework). Les données RDF peuvent être interrogées par des requêtes en utilisant Sparql qui est un langage spécifique conçu pour ce type de données.

A la différence de la plupart des bases de ressources, qui couvrent des domaines spécifiques, Dbpedia se nourrit de Wikipedia en exploitant des ressources gigantesques qui évoluent au fur et à mesure et qui couvrent tous les domaines. Dbpedia est disponible en 125 langues. Toutes ces versions linguistiques décrivent 38.3 millions d'entités. Au total, Dbpedia 2014 se compose de 3 milliards[2] de triplets RDF. Le tableau ci-dessous indique le nombre des instances pour plusieurs classes au sein de l'ontologie de Dbpedia.

---

[1] http://wiki.dbpedia.org/about. Consulté le 02/04/2015

[2] http://wiki.dbpedia.org/services-resources/ontology. Consulté le 02/04/2015

Tableau 1.1 Instances par classes en Dbpedia[3]

Class	Instances
Resource (overall)	4,233,000
Place	735,000
Person	1,450,000
Work	411,000
Species	251,000
Organisation	241,000

1.3.2 Types de données Dbpedia

Les triplets RDF de Dbpedia couvrent plusieurs domaines, tels que les données géographiques, films, musiques, livres, publications scientifique, etc. Dbpedia définit pour chaque instance un identifiant unique URI (*Uniform Resource Identifier*). Parmi les types de données extraites de Wikipedia par Dbpedia on trouve (Morsey et al.) (2012)

- *Les étiquettes (Labels)*: Chaque article dans Wikipedia a un titre qui est considéré par Dbpedia comme l'étiquette (***rdf :label***) d'une ressource qui correspond à cet article.

- *Les images :* Chaque image dans wikipedia liée à une ressource se réfère à Wikimedia Commons[4] : Elle est spécifiée par la propriété ***foaf:depiction*** dans Dbpedia.

---

[3] http://wiki.dbpedia.org/about. Consulté le 02/04/2015

[4] https://commons.wikimedia.org/wiki/Commons:Bienvenue?uselang=fr. Consulté le 08/04/2015

- *Les résumés (Abstracts)* : deux types d'abstract :
    - *Court résumé (short Abstract) :* C'est le premier paragraphe de chaque article Wikipedia. Il est défini dans Dbpedia par la propriété ***rdfs:comment.***

    - *Long résumé* (*long Abstract*) : Chaque texte écrit avant la table du sommaire est considéré comme un long résumé (*long abstract*). Il est défini dans Dbpedia par la propriété ***dbpedia-owl:abstract.***
- *Les liens interlangues (interlanguage links) :* Ce sont les liens qui lient les articles qui ont les mêmes sujets mais en différentes langues. Dbpedia les utilise pour attribuer des étiquettes et des résumés à la ressource en différentes langues.

- *Redirections (Redirects) :* Afin d'identifier les synonymes d'un terme dans un article, Wikipedia peut diriger vers d'autres articles. Dbpedia extrait ces redirection pour les utiliser en tant que références pour ses ressources (***dbpedia-owl:wikiPageRedirects***).

- *Infoboite (Infobox[5]) :* Il s'agit d'une table de données, située dans l'angle haut droite d'un article dans Wikipedia. Elle présente généralement un sommaire des informations importantes dans l'article.

- *Liens externes (External Links)* : Ce sont des liens qui contiennent des références vers d'autres ressources sur le Web. Dbpedia les définit par ***dbpedia:reference.***

- *Pages d'accueil (HomePages) :* Ce sont les liens vers les pages d'accueil des entités telles que les entreprises et les organisations. Définis en Dbpedia par ***foaf:homepage.***

---

[5] http://en.wikipedia.org/wiki/Help:Infobox. Consulté le 08/04/2015

- *Données personnelles (Person Data) :* Sont les données personnelles sur les personnes, telles que le nom de famille et la date de naissance, qui sont définies respectivement en Dbpedia par ***foaf:surname*** et ***dbpedia:birthDate.***

- *Catégories (Categories) :* Les articles de Wikipedia sont disposés dans des catégories en utilisant les vocabulaires SKOS[6].

1.3.3 La structure de Dbpedia

Les articles dans Dbpedia contiennent divers types de données structurées sous la forme de la syntaxe wiki. Ces données comprennent plusieurs types, comme mentionné ci-dessus (infobox, images, liens externes, ...etc.). La figure 1.3, donne un aperçu sur la structure de Dbpedia qui permet l'extraction des informations afin de construire une base de connaissances.

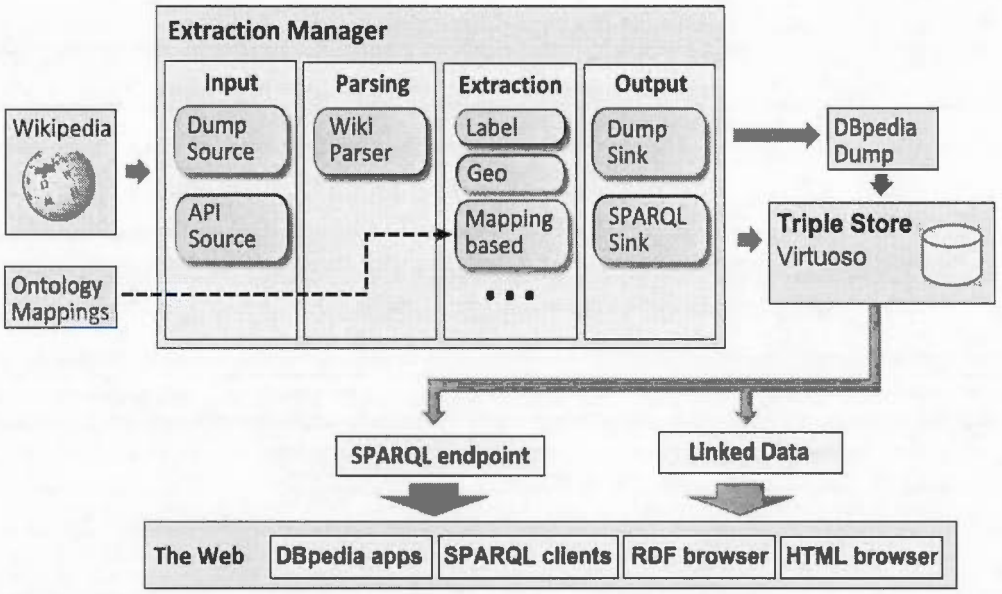

Figure 1.3 Aperçu de la structure de Dbpedia (Lehmann et al., 2014).

[6] http://www.w3.org/2004/02/skos/. Consulté le 08/04/2015

Dbpedia est structuré comme suit (Lehmann et al.) (2014) :

- *L'entrée (Input)* : Le cadre d'extraction Dbpedia est mis en place pour gérer deux types de flux :
    - L'extraction à base de vidage régulier (*Dump Source*) : Une mise à jour de la base de connaissances de Dbpedia se fait chaque mois en prenant comme source la base de données de Wikipedia.
    - L'extraction en direct : Dbpedia prend comme source directe MediaWiki[7] en utilisant une API (MediaWiki API[8]) qui permet l'extraction de ces données.
- L'analyse (*parsing*) : Chaque page de Wikipedia est analysée par le parseur de Wiki, qui transforme son code source en un arbre syntaxique abstrait.
- L'extraction : Le cadre d'extraction Dbpedia emploie divers extracteurs pour traduire les arbres syntaxiques abstraits en déclarations RDF.
- La sortie (*Output*) : Une fois les déclarations RDF extraites, Dbpedia propose plusieurs formats d'écriture pour ces déclarations tels que : RDF/XML, N-Triples, Turtle, etc. Dans notre projet nous allons utiliser le Format RDF/XML.

---

[7] https://wikimediafoundation.org/wiki/Accueil. Consultée le 15/04/2015

[8] https://www.mediawiki.org/wiki/MediaWiki. Consultée le 15/04/2015

1.3.4 Les accès aux données Dbpedia

Il existe trois façons pour interroger les ressources sur Dbpedia (Auer et al.) (2007):

- *Linked data*: C'est une méthode de publication des données RDF sur le web en utilisant les URIs pour identifier les ressources, HTTP pour récupérer les descriptions des ressources et le modèle RDF pour les structurer et les lier.

- *Le protocole SPARQL Endpoint* : Dbpedia dispose de ce protocole pour offrir la possibilité de récupérer les ressources directement en ligne en envoyant des requêtes sur *http://dbpedia.org/sparql* hébergé par le serveur VIRTUOSO UNIVERSAL[9]. *SPARQL Endpoint* va être expliqué en détails dans les chapitres qui suivent.

- *RDF DUMP*: Un autre moyen d'extraire les ressources RDF à partir de Dbpedia hors ligne consiste à télécharger RDF DUMP qui permet de parcourir la base de données Dbpedia sans avoir besoin d'être connecté à internet.

[9] http://virtuoso.openlinksw.com. Consultée le 21/05/2015

1.3.5 Présentation des classes Dbpedia

L'ontologie Dbpedia a été créée en se basant sur une table de données spécifique qu'est l'Infobox, en couvrant actuellement 685 classes décrites par 2795 propriétés. La figure 1.4 montre un exemple sur l'extraction d'une sémantique d'une Infobox sur la commune d'Oran en Algérie.

```
{{Infobox Commune d'Algérie
| nom            = Oran
| nomAlgérien    = {{lang|rtl|ar|وهران}} (Wahran)
| image          = Oran_facade_maritime.JPG
| légende        = Vue d'Oran depuis Santa Cruz
| blason         = Armoirie oran.png
| drapeau        =
| région         = [[Oranie]]
| wilaya         = Oran
| daïra          = Oran
| chef-lieu      = Oran
| ons            = 3101
| cp             = 31000
| tel            = 041
| président apc  = Saddek Benkada
| mandat         = [[2007]]-[[2012]]
| budget         = 4,8 milliards de [[Dinar algérien|DA]] en 2008
| population     = 1200000
| année_pop      = 31-12-07
| gentilé        = Oranais(es)
| patrons        = [[Sidi el-Houari]]
| fête patronale =
| latitude       = 35.691
| longitude      = -0.642
| position       =
| alt mini       =
| alt maxi       =
| superficie     = 2121
| code_cadastre  =
| site_web       = http://www.oran-dz.com/
}}
```

Figure 1.4 Infobox sur la commune d'Oran en Algérie.

Prenons comme exemple la classe Artist sur Dbpedia qu'on peut voir par :

- Sa page de mappings
  `http://mappings.dbpedia.org/server/ontology/classes/Artist`
  où nous pouvons voir ses propriétés telles que *careerStation* de domaine *Person* et range *careerStation*. Cette propriété fait le lien vers une étape dans la carrière d'une personne. La figure 1.5 montre un extrait de la classe *Artist* représentée par sa page de mappings.

Artist (Show in class hierarchy)

Label (be): артыст
Label (it): artista
Label (pl): artysta
Label (ga): ealaíontóir
Label (el): καλλιτέχνης
Label (ko): 예술가
Label (ja): 芸術家
Label (en): artist
Label (ru): художник
Label (nl): kunstenaar
Label (de): Künstler
Label (fr): artiste
Super classes: Person

Properties on Artist:

Name	Label	Domain	Range	Comment
academyAward (edit)	Academy Award	Artist	Award	
achievement (edit)	achievement	Person	owl:Thing	
activeYear (edit)	active year	Person	xsd:string	
activeYears (edit)	active years	Person	xsd:string	Also called "literal". Use this if the active years are in one field that can't be split. Else use activeYearsStartYear and activeYearsEndYear
activeYearsEndDateMgr (edit)	active years end date manager	Person	xsd:string	
activeYearsEndYearMgr (edit)	active years end year manager	Person	xsd:gYear	
activeYearsStartDateMgr (edit)	active years start date manager	Person	xsd:date	
activeYearsStartYearMgr (edit)	active years start year manager	Person	xsd:gYear	

Figure 1.5 Extrait de la page mappings de la classe Artist.

- Sa page de l'ontologie *http://dbpedia.org/ontology/Artist* qui est sous-classe de *Person* et super-classe de *Actor, Comedian, Writer, Dancer,* etc. Cette classe est décrite par des propriétés qui lui sont propres, comme *AcademyAward,* de domaine *Artist* et de range *Award.* Elle hérite de tout ce qui vient de ses super-classes par exemple, on ne retrouve pas *age* et *Agent*. La figure 1.6 montre un extrait de la classe *Artist* représentée par sa page de l'ontologie.

Property	Value
rdf:type	• owl:Class
rdfs:isDefinedBy	• http://dbpedia.org/ontology/
rdfs:label	• artist • artiste
rdfs:subClassOf	• dbo:Person
owl:equivalentClass	• wikidata:Q483501
wdrs:describedby	• dbo:data/definitions.ttl
http://www.w3.org/ns/prov#wasDerivedFrom	• http://mappings.dbpedia.org/index.php/OntologyClass:Artist
is http://open.vocab.org/terms/defines of	• http://dbpedia.org/ontology/
is http://open.vocab.org/terms/describes of	• dbo:data/definitions.ttl
is rdfs:domain of	• dbo:academyAward • dbo:aflAward • dbo:associatedAct • dbo:baftaAward • dbo:cesarAward • dbo:disciple • dbo:dutchFKDCode • dbo:emmyAward • dbo:field • dbo:filmFareAward • dbo:gaudiAward • dbo:goldenGlobeAward • dbo:goyaAward • dbo:grammyAward • dbo:instrument • dbo:mbaId • dbo:mentor • dbo:movement • dbo:polishFilmAward • dbo:style • dbo:tonyAward • dbo:training • dbo:voiceType
is rdfs:range of	• dbo:artPatron • dbo:associatedAct • dbo:disciple • dbo:mentor

Figure 1.6 Extrait de la page de l'ontologie de la classe Artist.

1.4 Analyse formelle de concepts (AFC) :

L'analyse formelle de concepts (AFC) offre une représentation formelle des données sous forme d'une structure conceptuelle (Ganter et Wille) (1999) et permet ainsi de les analyser. AFC prend en compte les attributs et les objets en formant un tableau binaire de deux dimensions. Elle fournit un cadre mathématique de construction hiérarchique de concepts à partir ce qu'on appelle un contexte formel (Dau et Klinger) (2005).

1.4.1 Contexte formel

Définition 1 (Contexte formel) :

Un contexte formel est un triplet $K = (O, A, I)$ où O est un ensemble d'objets, A un ensemble d'attributs et I une relation d'incidence entre O et A. (Hacene et al.) (2008).

Tout au long de ce chapitre, nous allons utiliser comme exemple un ensemble de données sur des corps célestes classés dans la base de données de SIMBAD[10], tiré de (Hacene *et al.*) (2008). Dans le but de comprendre le comportement de l'univers, le Centre de données astronomiques de Strasbourg s'occupe de la classification des corps célestes. Cette tâche est réalisée manuellement en se basant sur les propriétés de chaque corps afin de les associer à des classes. Avec un nombre gigantesque de corps célestes, les experts trouvent que la méthode manuelle de classification manque de précision d'où la nécessité d'une nouvelle approche.

Exemple 2 :

Dans le tableau 1.2, (Hacene et al.) (2008) présentent un moyen de construire une ontologie sur les corps célestes observés.

---

[10] http://simbad.u-strasbg.fr/simbad/. Consultée 23/03/2015.

Tableau 1.2 Contexte formel binaire des corps célestes.

Celestial bodies						
	Emitting	*Accreting*	*Collimating*	*Observed*	*Located*	*Grouping*
PSRA	X			X	X	
NGC3570				X		X
Andromeda		X		X		X
M87			X	X	X	
HR2				X	X	
NGC2018			X	X	X	
HR5223	X			X	X	
SS433	X			X	X	

Le Tableau 1.2 montre les liens entre les corps célestes (objets *O*) et leurs propriétés (attributs *A*). Ce tableau représente un contexte formel nommé $K_a = (O, A, I)$ avec :

O= {PSRA, NGC3507, Andromeda, M87, HR2, NGC2018, HR5223, SS433} qui sont quelques corps célestes observés par la station SIMBAD.

A= {Emitting, Accreting, Collimating, Observed, Located, Grouping}, sont les actions effectuées par les satellites sur les corps célestes :

Emitting : Diffusion des signaux émis par les corps célestes et interceptés par les satellites.

Accreting : Observation des matières accumulées sur les surfaces des corps célestes.

Collimating : Orientation des satellites qui permet l'obtention des faisceaux de rayons de lumière parallèles à partir des corps célestes.

Observed : Observation des corps célestes par les satellites.

Located : Localisation des corps célestes par les satellites.

Grouping : Regroupement des corps célestes dans des catégories par les satellites.

I représente les relations entre les corps célestes et les actions.

Dans notre exemple le corps céleste PSRA a été observé et localisé après la réception d'un signal. Ces informations sont déclarées dans le tableau précédent par les croix qui lient le corps céleste à ces actions.

1.4.2 Opérateurs de dérivation

Définition 2 (Opérateurs de dérivation) :

Soient *O, A, I, K* tels que décrits à la définition 1 et, $X \subseteq O$. On définit l'opérateur de dérivation sur *X* comme suit. $X' = \{a \in A \mid oIa \ \forall o \in X\}$. De même, on définit l'opérateur de dérivation sur $Y \subseteq A$ comme suit. $Y' = \{o \in O \mid oIa \ \forall a \in Y\}$. (Hacene *et al.*) (2008)

Exemple 3 :

Supposons *O, A* et *K* tels que décrits dans l'exemple 1 et, $X \subseteq O$ et $Y \subseteq A$.

On obtient les dérivations suivantes :

(1) *X = {M87,NGC2018}. X' = {M87,NGC2018}' = {Collimating, Observed, Located}.*

(2) *Y = {Collimating, Observed, Located).Y'= {Collimating, Observed, Located}'= {M87,NGC2018}.*

(3) *Y = {Accreting, Grouping}. Y' = {Accreting, Grouping}' = {Andromeda}.*

(4) *X = {Andromeda}. X' = {Andromeda}' = {Accreting, Observed, Grouping}.*

(5) *Y = {Accreting, Observed, Grouping}, Y' = {Accreting, Observed, Grouping}' = {Andromeda}.*

Dans cet exemple, le calcul de la dérivation de Y = {Accreting, Grouping} produit un X = {Andromeda}. Le calcul de cette dérivation a produit un ensemble fermé[11] X dans O. En appliquant la dérivation sur X = {Andromeda}, on obtient Y = {Accreting, Observed, Grouping}. Cette fois-ci, le calcul de la dérivation sur X a produit un ensemble fermé $Y \subseteq A$. La dérivation de ce dernier a produit la même valeur X = {Andromeda} que calculé dans la ligne (3) de l'exemple. Cette observation permet de définir la notion de concept formel dans AFC.

1.4.3 Concept formel

Définition 3 (Concept formel) :

Soient O, A, K tels que décrits à la définition 1, $X \subseteq O$ *et* $Y \subseteq A$. Un concept formel est une paire (X, Y) ayant $X' = Y$ *et* $Y' = X$. Dans ce cas, X est appelé l'extension et Y est appelé l'intension (Wille, 2009).

---

[11] Plus d'informations sur le sujet de la fermeture des ensembles sont disponibles à l'adresse : *http://www.hypertextbookshop.com/dataminingbook/public_version/contents/chapters/chapter002/section004/blue/page002.html*

Exemple 4 :

Supposons *O, A* et *K* tels que décrits dans l'exemple 1.

La paire *({M87,NGC2018}, {Collimating, Observed, Located})* est-elle un concept formel ?

Le calcul des dérivations donne :

X' = {Collimating, Observed, Located} et Y' = {M87,NGC2018}.

Alors *({M87,NGC2018},{Collimating, Observed, Located})* est un concept formel.

La paire *({Andromeda}, {Accreting , Grouping})* est-elle un concept formel?

Le calcul des dérivations donne :

X' = {Accreting, Observed, Grouping} et Y' = {Andromeda}.

Donc *({Andromeda}, {Accreting, Grouping})* n'est pas un concept formel, par contre *({Andromeda}, {Accreting, Observed, Grouping})* est un concept formel.

1.4.4 Treillis de concepts formels

Un concept formel peut être représenté dans un treillis de concepts dans lequel chaque concept comprend un ensemble d'objets et un ensemble d'attributs connexes. Les concepts formels formant ce treillis sont ordonnés hiérarchiquement par une relation d'inclusion entre les concepts.

Définition 4 (Treillis de concepts formels) :

Soit le contexte formel $K= (O, A, I)$, $C_1= (A_1, B_1)$, $C_2= (A_2, B_2)$ deux concepts formels, une relation hiérarchique *Subconcept-Superconcept* est définie par :

$(A_1, B_1) \leq (A_2, B_2) <=>- A_1 \subseteq A_2 (<=> B_1 \supseteq B_2)$. Wille (2009)

La figure 1.7 montre un treillis construit avec Galicia qui représente les contextes formels du tableau 1.2

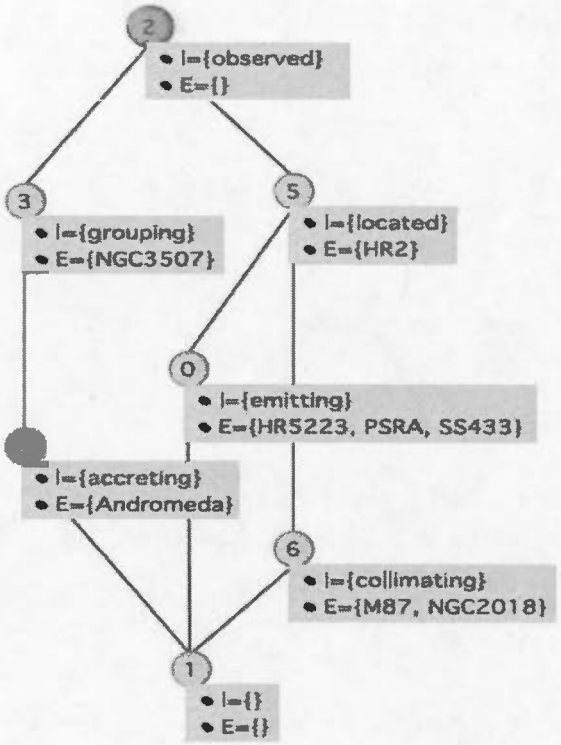

Figure 1.7 Treillis des contextes formels du corps célestes.

1.4.5 Règle d'association

Définition 5

Soit $I = \{i_1, i_2, ..., i_m\}$ un ensemble d'items. Soit $T = \{t_1, t_2, ..., t_n\}$ un ensemble de transactions telles que t_i sont-sous-ensembles de I. Une règle d'association s'exprime sous la forme :

$Y_1 \rightarrow Y_2$,où $Y_1 \in T$, $Y_2 \in T$ et $Y_1 \cap Y_2 = \emptyset$ (Godin, Missaoui) (1994)

1.4.6 Règle d'implication dans un contexte formel :

Définition 6

Soit le contexte formel $K= (O, A, I)$ et $Y_1, Y_2 \subseteq A$ deux ensembles d'attributs. On dit que Y_1 implique Y_2 si et seulement tout objet de O qui a les attributs de Y_1 a aussi les attributs de Y_2 :

$Y_1 \rightarrow Y_2$ *ssi* $Y_1' \subseteq Y_2'$ autrement dit : $Y_1 \rightarrow Y_2$ *ssi* $Y_2 \subseteq Y_1''$ (Ganter et Wille) (1999)

Exemple 5 :

Dans le contexte formel précédent des corps célestes, on a l'exemple d'implication suivant :

$Y_1 = \{Emitting\} \rightarrow Y_2 = \{Located\}$ car $Y_1 = \{Emitting\}$ est possédé par les objets $Y_1' = \{PSRA, HR5223, SS433\}$ et $Y_2 = \{Located\}$ est possédé par les objets de $Y_2' = \{PSRA, HR5223, SS433, M87, NGC2018, HR2\}$.

Donc on voit bien que $Y_1' \subseteq Y_2'$. Par définition on observe que dans le treillis des contextes formels du corps céleste l'attribut *Emitting* introduit par le concept C0 implique l'attribut *Located* introduit par le super-concept C5.

1.4.7 Le support et la confiance Godin (2006)

Définition 7 (Support)

Supposons R un ensemble d'items, r une base de données sur R et $Y \subseteq R$ un itemset. On note $M(r, Y) = \{t \in r \mid Y \subseteq t\}$, le multiset de transactions qui contiennent Y. Le support de Y sur M est noté $support(M, Y) = |M(r, Y)|$ et signifie le nombre de transactions dans M correspondant à Y.

Le support d'une règle d'association $Y_1 \Rightarrow Y_2$ se définit comme suit :

$$support(r, Y_1 \Rightarrow Y_2) = support(r, Y_1 \cup Y_2).$$

Définition 8 (Confiance)

Supposons R un ensemble d'items, r une base de données sur R et $Y \subseteq R$ un itemset. La fréquence de Y est définie de la façon suivante : $f(r, Y) = support(r, Y)/|r|$. En d'autres termes, la fréquence n'est autre que le support en notation relative.

La confiance d'une règle d'association $Y_1 \Rightarrow Y_2$ se définit comme suit :

$$confiance(r, Y_1 \Rightarrow Y_2) = support(r, Y_1 \Rightarrow Y_2) / support(r, Y_1).$$

1.4.8 Contextes formels à valeurs multiples

Dans des données réelles les attributs peuvent avoir plusieurs valeurs, qui donnent une autre dimension au contexte formel. Les contextes avec ces attributs sont les contextes à valeurs multiples *(multi-valued)*.

Définition 7 (Contextes formels à valeurs multiples) :

Un contexte formel à valeurs multiples est un quadruplet (G, M, W, I) où G est un ensemble d'objets, M est un ensemble d'attributs à valeurs multiples, W est l'ensemble de valeurs prises par les attributs et $I \subseteq G \times M \times W$ une relation ternaire entre G, M et W telle que

$$(g, m, w) \in I \text{ et } (g, m, v) \in I \text{ implique } w = v$$

Les notations $(g, m, w) \in I$ et $I(g, m) = w$ sont équivalentes et expriment que l'attribut m a la valeur w pour l'objet g. Messai (2009).

Exemple 6 :

Dans cet exemple, un contexte formel à valeurs multiples a été construit à partir des données récoltées par des télescopes.

Tableau 1.3 Contexte à valeurs multiples des télescopes d'observation.

Télescopes			
	Perigee	*OrbitalPeriod*	*Mass*
BeppoSAX	600 km	96 min	1400 kg
XMM-Newton	114000 km	48 hours	3800 kg
Chandra	26300 km	66 hours	1790 kg

Le tableau 1.3 (Hacene et al.) (2008) représente trois télescopes avec leurs principales caractéristiques telles que la période orbitale et la masse. Dans cet exemple les attributs ont des valeurs multiples, comme par exemple la période orbitale avec les valeurs : (96 min, 48 heures, 66 heures).

1.4.9 Binarisation des contextes à valeurs multiples

Une technique pour analyser des contextes à valeurs multiples consiste à les transformer en des contextes à valeurs uniques en appliquant une méthode spécifique appelée *graduation (Scaling) des attributs.* La méthode *Scaling* transforme les attributs a non binaires du contexte formel en des attributs binaires en utilisant un contexte d'échelle $K_a = (V_a, P_a, J_a)$ où P_a sont les abstractions des valeurs de V_a *et* J_a est la relation ternaire.

Par exemple, l'attribut *Perigee* dans le tableau 1.3 peut avoir deux valeurs, il peut être substitué par deux attributs représentant ses valeurs. Soient *Low* ou *High.* Chacun d'eux est exprimé comme un prédicat (ex. : *Perige* $\leq$ *1000* km est considéré comme *Low).* Même chose pour les attributs *OrbitalPeriod* et *Mass* qui ne peuvent avoir respectivement que les valeurs (*Short* et *Long*) et (*Heavy* et *Lightweight*). Le tableau 1.4 représente les contextes formels à valeurs uniques après la binarisation.

Tableau 1.4 Contextes formels des télescopes d'observation après la binarisation (Hacene *et al.)* (2008).

Télescopes						
	Perigee		*OrbitalPeriod*		*Mass*	
	High	*Low*	*Long*	*Short*	*Heavy*	*Lighweight*
BeppoSax		X		X		X
XMM-Newton	X		X		X	
Chandra	X		X			X

A partir des contextes formels à valeurs unique, on obtient le treillis de la figure 1.8

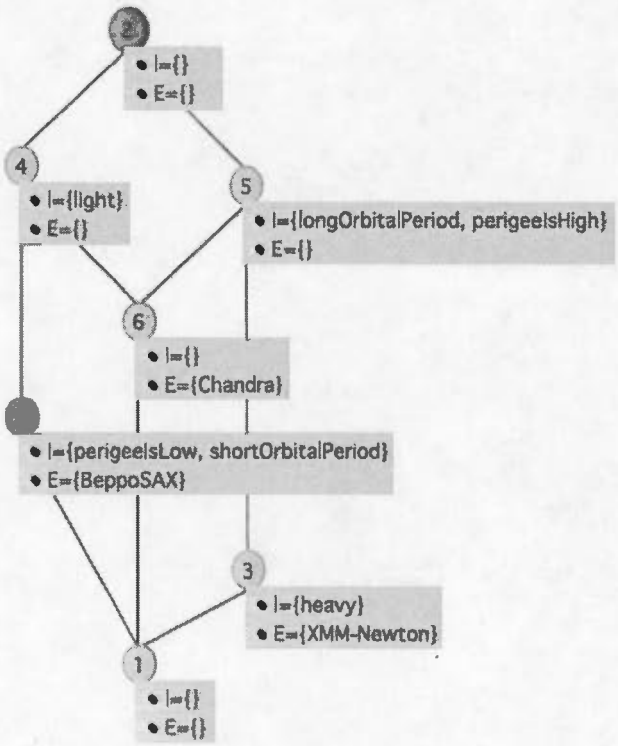

Figure 1.8 Treillis des contextes formels des télescopes (Hacene *et al.)* (2008).

1.5 Analyse relationnelle de concepts (ARC)

L'analyse relationnelle de concepts (ARC) est une extension de l'analyse formelle de concepts (AFC). Elle a été conçue pour répondre au besoin de prendre en compte les relations entre les objets de différents contextes. Cette méthode d'analyse de données génère des groupes de concepts à partir de contextes multi-relationnels (Rouane-Hacene *et al.)* (2013). Elle repose sur les principaux points suivants :

- Un modèle relationnel basé sur un modèle entité-relation.
- Des attributs relationnels ajoutés au contexte source des relations.
- Un processus itératif de conception d'un treillis de concepts jusqu'à l'obtention d'un treillis final complet qui ne peut plus évoluer par l'ajout des attributs relationnels.

1.5.1 Contexte relationnel

ARC a comme entrée une structure appelée famille de contextes relationnels (FCR) qui est composée d'une collection de contextes standards $K = \{K_i\}_{i=1,\ldots,n}$ et d'un ensemble de relations binaires inter-contextes $R = \{r_k\}_{k=1,\ldots m}$, avec $r \in R$ qui relient les objets appartenant aux différents contextes K.

Définition 7 (Famille de contextes relationnels (FCR))

Une FCR est une paire (K, R) où K est un ensemble de contextes $K_i = (O_i, A_i, I_i)$ et R = $\{r_k\}$ un ensemble de relations $r_k \subseteq O_i \times O_j$, avec Oi les objets de l'ensemble K_i (domaine de r_k) et O_j les objets de l'ensemble K_j (co-domaine de r_k). (Hacene *et al.*, 2008)

Exemple 8 :

Prenons par exemple le concept C_0 de la figure 1.7 représentant les corps célestes HR5223 et SS433 qui sont observés par le télescope BeppoSAX représenté par C_0 de la figure 1.8. De cette façon on peut définir les relations suivantes entre les télescopes et les corps célestes : *<observé par Xray>* raccourcie à *(OWxray)* et *<observé par Infrared>* raccourcie à *(OWinfrared.)*. Ces deux contextes relationnels sont présentés dans les tableaux 1.5 et 1.6 :

Tableau 1.5 Relation OWinfrared.

OWinfrared			
	BeppoSAX	XMM-Newton	Chanda
HR5223	X		
SS4333	X		

Tableau 1.6 Relation OWxray.

OWxray			
	BeppoSAX	XMM-Newton	Chanda
M87		X	
NGC2018			X

Afin de construire le contexte relationnel, on applique le mécanisme de graduation *(scaling)* existentielle. Donc les relations sont interprétées comme des attributs dont les valeurs sont les ensembles d'objets. Nous définissons d'abord le domaine (*dom*), le co-domaine (*ran*) d'une fonction à partir de la FCR et la fonction *rel(K)* et enfin *la graduation existentielle (existential Scaling)*.

Définition 9 (domaine et co-domaine)

(K, R) est un FCR, il existe une paire de fonctions (*dom* et *ran*) qui associe les relations de R aux domaines et aux co-domaines respectifs correspondant aux ensembles *O* d'objets de K. Avec $O = \{O|K = (O, A, I) \in K\}$. (Rouane-Hacene et al.) (2013)

- La fonction domaine, notée *dom*, est telle que :

$R \rightarrow O$ *avec* $dom(r) = O_{i1}$ *si pour tous* $(x, y) \in r, x \in O_{i1}$

La fonction retourne les objets du contexte source de la relation qui participent à la relation.

Exemple: $dom\ (OWxray) = O_p = \{M87, NGC2018\}$

- La fonction range est *ran* telle que:

$R \rightarrow O$ *avec* $ran(r) = O_{i2}$ *si pour tous* $(x, y) \in r, y \in O_{i2}$

La fonction *ran* retourne tous les objets du contexte destination qui participent à la relation.

Exemple: $ran\ (OWxray) = O_D = \{BeppoSAX, XMM\text{-}Newton, Chanda\}$

Définition 10 (Fonction *rel(K)*) :

La fonction *rel(K)* permet de retrouver toute la famille de relations initiées par un contexte (*K*). Cette fonction est définie par :

$$rel : K \rightarrow \rho(R) \text{ avec } rel\ (K = (O, A, I)) = \{r \in R |\ dom(r) = O\}.$$

K est l'ensemble des contextes et R est l'ensemble des relations de cette famille des contextes. (Rouane-Hacene *et al.*, 2013).

Définition 11 (*Existential Scaling*)

Soit le contexte $K_i = (O_i, A_i, I_i)$ et la relation $r \subseteq O_i \times O_j$, et le treillis $\mathcal{L}_j$ de K_j. l'image de K_i pour l'opérateur d'échelle existentielle $sc_\exists$ est :

$$sc_\exists (K_i) = (O_i, A_i^+, I_i^+) \text{ où}$$

$A_i^+ = A_i \cup \{r : c | c \in \mathcal{L}_j\}$:Un ensemble de nouveaux attributs est ajouté, un par concept

$I_i^+ = I_i \cup \{(o, r : c) | o \in \mathcal{L}_j, r(o) \cap Ext(c) \neq \emptyset\}$: Les nouveaux attributs sont affectés aux objets du K en respectant la discipline associée à l'opérateur.

Exemple 8:

Supposons que les corps célestes dans le treillis de la figure 1.8 sont mis à l'échelle par la relation *OWxray*. Alors *OWxray (NGC2018)* = *{Chandra}* et le télescope *Chandra* aux extensions des concepts c_2, c_4, c_5 et c_6. Le contexte des corps célestes est prolongé par les attributs relationnels de la forme : $r{:}c_i$ où $i=\{2, 4, 5, 6\}$.

Le tableau 1.7 présente la graduation des deux relations *OWxray* et *OWinfrared* et de leur intégration dans le contexte des corps célestes sous la forme de nouveaux attributs relationnels.

Tableau 1.7 Graduation du contexte des corps célestes par les relations *OWxray* et *OWinfrared.*

	OWxray : c_0	OWxray : c_1	OWxray : c_2	OWxray : c_3	OWxray : c_4	OWxray : c_5	OWxray : c_6		OWinfrared : c_0	OWinfrared : c_1	OWinfrared : c_2	OWinfrared : c_3	OWinfrared : c_4	OWinfrared : c_5
HR5223									X		X		X	
M87			X	X		X								
SS433									X		X		X	
NGC2018			X		X	X								

Note : les corps célestes qui n'ont pas été affectés par la graduation n'ont pas été mentionnés dans le tableau 1.7.

Avec le contexte relationnel du tableau 1.7, on obtient le treillis de la figure 1.9 :

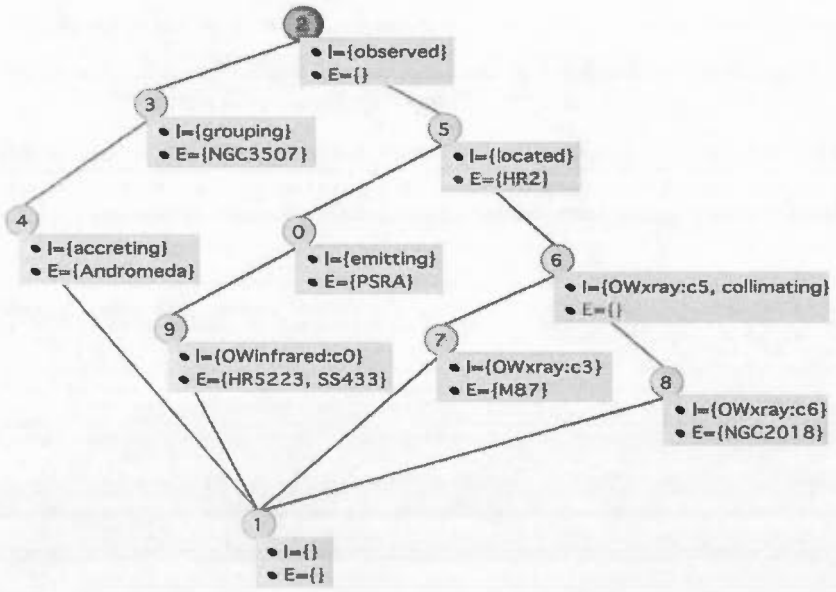

Figure 1.9 Treillis final des corps célestes.

Dans le treillis final des corps célestes, on constate l'ajout des nouveaux concepts. Par exemple le c#6 qui représente les corps céleste *M87* et *NGC2018* reconnus comme des étoiles binaires par l'expert, car ils sont observés et collimatés.

1.6 Notre approche

Notre approche étudie l'ontologie Dbpedia pour observer les usages et améliorer les pratiques, en utilisant les régularités existantes entre les instances de son jeu de données, afin de construire un schéma bien identifié à l'aide de l'analyse relationnelle de concepts ARC.

L'approche a comme entrée un ensemble de ressources Dbpedia avec leurs propriétés (y compris les ***rdf :type*** qui les lient à des classes de l'ontologie) et comme sortie, un schéma qui reflète mieux la distribution des propriétés sur les ressources et une liste de co-occurrences de propriétés (les plus significatives) qui peuvent supporter la saisie et la correction des anomalies dans les données.

Nous avons choisi une approche basée sur l'ARC pour pouvoir raffiner aussi bien l'ensemble des classes que l'ensemble des propriétés. Cela exige également que tous les éléments d'un graphe RDF, les sujets et les objets, mais surtout les prédicats, soient traduits comme des objets formels en ARC. Cette façon de faire se distingue de toutes les études précédentes comme nous montrons dans le chapitre suivant.

CHAPITRE II
ETAT DE L'ART

L'accroissement des données sur le web a provoqué la création d'un grand nombre de sources de données. Ces sources couvrent tous les domaines pour publier des données structurées sur le web selon les principes et les technologies du web sémantique, en particulier en utilisant le modèle RDF.

Un modèle RDF représente l'information sur le web par des graphes orientés où chaque nœud est un objet alors que les arcs sont les relations entre ces objets. Par conséquent, les graphes RDF constituent une simple mais puissante façon de représenter les descriptions conceptuelles des données de web. Représenter les données sur le web en adoptant un tel format standard rend l'exploitation et la gestion de ces données plus facile et offre aux applications basées sur des données liées (*linked data*) la possibilité de naviguer à travers ces données. Cela permet la construction de schémas pour les données liées en exploitant les treillis de concepts générés à partir des données RDF (Mehri-Dehnavi) (2014).

Dans ce chapitre, nous passons en revue certaines approches d'analyse des données liées. Ces approches ont pour but la complétion des connaissances par une extraction à partir des données présentes sur le web. Deux sortes de complétion sont traitées : Complétion de schéma et complétion de données.

2.1 Complétion de schéma

L'ontologie de Dbpedia que nous avons choisi d'étudier contient assez peu de classes. Chaque classe définit un grand nombre d'attributs, mais les instances (contrairement à des instances dans un langage de programmation par objets) ne donnent pas une valeur à toutes les propriétés. L'ontologie est en ce sens une description de possibles, mais pas forcément de la réalité des objets décrits.

Lorsque l'on utilise un ensemble d'instances et que l'on nettoie leurs valeurs (primitives) en construisant des descripteurs, on se retrouve avec une sorte de schéma. Pour faire référence

à un modèle de classes, les descripteurs peuvent être assimilés à des classes et les propriétés entre descripteurs correspondent à des associations.

De ce fait, notre travail a des proximités avec les travaux utilisant l'Analyse formelle de concepts, l'Analyse Relationnellè de concepts et d'autres techniques pour normaliser un modèle de classes en faisant apparaître de nouvelles abstractions.

Fennouh *et al.* (2015) proposent une approche basée sur l'analyse relationnelle de concepts pour la restructuration d'ontologie en utilisant une méthode de filtrage combinant des métriques de pertinence et un algorithme basé sur des règles heuristiques. Cette approche comprend les étapes suivantes :

1- *L'alignement* : Compare les éléments de l'ontologie initiale afin de trouver des similarités.
2- *Le codage* : Transforme l'ontologie initiale en une unique famille de contextes relationnels FCR. Chaque contexte relationnel représente une catégorie d'éléments du méta-modèle de l'ontologie.
3- *L'analyse* : Les treillis initiaux sont construits avec l'AFC et les relations inter-contexte transformées à des attributs relationnels. Les treillis finaux sont construits selon le processus itératif de l'ARC.
4- *Le filtrage* : Élague les concepts formels inintéressants en basant sur trois métriques : *la stabilité*, *la densité* et *la similarité sémantique*.
5- *Le codage inverse* : Convertit les concepts formels du treillis filtrés considérés comme pertinents à des éléments ontologiques au format OWL (*Web Ontology Langage*).

Mehri-Dehnavi (2014) Propose une approche basée sur l'analyse de concepts formels, qui aide à apprendre le niveau du schéma à partir du niveau instance en extrayant la structure conceptuelle des données RDF. Cette approche comprend trois étapes :

1- *Convertir RDF à une entrée pour AFC* : Dans cette étape, les ressources et leurs propriétés sont extraites des données RDF afin d'être utilisées pour construire la table de contexte formel. Un outil AFC (Galicia) génère le treillis de concepts formels à partir de ce contexte formel.
2- *Convertir la sortie AFC (Treillis) à un schéma RDF*: Cette étape consiste à convertir le treillis de concepts obtenu dans l'étape précédente à un graphe RDFS selon trois règles :
    a. Règle de nœud : Tous les nœuds sauf ceux qui ont tous les objets.
    b. Règle *rdfs:subclass* représente la relation entre les classes. Elle construit selon la hiérarchie entre les concepts de treillis.
    c. Règle prédicat : Permet d'étudier toutes les propriétés liées aux classes.
3- *Attribuer des noms plausibles aux classes de RDFS* : Choisir des noms appropriés pour les classes dans le graphe RDFS.

Völker et Niepert (2011) proposent *Statical Schema Induction (SSI)*, une approche pour générer des ontologies à partir des jeux de données RDF. Cette approche est basée sur l'extraction des règles d'association afin de construire des ontologies. Elle se déroule en trois phases :

1- *Acquisition de la terminologie :* C'est la composition du vocabulaire de l'ontologie, en utilisant les tables des ensembles de données contenant les URIs de toutes les ressources RDF. Ces tables sont extraites en posant des requêtes SPARQL.
2- *Création de tables de transaction et l'extraction de règles d'association :* Consiste à créer des tables de transaction, ensuite trouver des règles d'association afin de concevoir les différents types d'axiomes qui vont faire partie de l'ontologie.
3- *Construction de l'ontologie* : En triant en premier temps les axiomes générés par ordre décroissant, en fonction de leurs valeurs de certitude. Ensuite les ajouter à

l'ontologie un par un, en vérifiant la cohérence de l'ontologie après l'ajout de chaque axiome.

Nikolov *et al.* (2010) proposent une approche d'adaptation du schéma. Leur approche cherche les relations au niveau schéma entre les données liées référentielles dans les instances disponibles et réutilise ces relations afin de faciliter la création de nouveaux liens de coréférence. Cette approche comprend les étapes suivantes :

1) *Clustering des individus identiques* : Tous les individus identiques dans un jeu de données sont groupés ensemble.

2) *Établir des relations entre les groupes et les termes de schéma* : Si un individu d'un groupe appartient à une classe spécifique, alors tout le groupe et toutes ces propriétés appartiennent à cette classe.

3) *Déduire les mappings* (correspondances) entre les termes de schéma.

Curé *et al.* (2013) Proposent une approche basée sur AFC, pour l'intégration des données. Cette approche prend une ontologie de web sémantique comme un schéma cible et une base de données NOSQL comme source. Cette approche comprend trois étapes :

1) La création d'une ontologie associée à chacune des sources de données.

2) L'alignement de ces ontologies.

3) La création d'une ontologie globale en tenant compte des correspondances.

2.2 Complétion de données

Cette section survole quelques approches qui ont pour but la complétion de données.

Dbpedia traite divers types d'informations qui sont présentées comme des classes, des instances et des relations. Wienand et Paulheim (2014) proposent une approche pour détecter les erreurs dans les attributs numériques primitives dans Dbpedia. En combinant plusieurs méthodes de détection des valeurs aberrantes et divers estimateurs de dispersion avec

différentes stratégies de prétraitement comme le regroupement (*clustering*) des sujets par des types simples et par des types de vecteurs. Cette approche comprend deux étapes : Premièrement, le regroupement des sujets par rapport leurs types. Deuxièmement, l'application des méthodes de détection des valeurs aberrantes à ces groupes.

1- *Regroupement des sujets* : Cette étape consiste à grouper les sujets de chaque propriété selon chaque type RDF trouvé dans l'ensemble de ces sujets. Cependant tous les types sont utiles pour le processus de détection des valeurs aberrantes.

2- *Détection des valeurs aberrantes* : La détection des valeurs aberrantes classiques suppose une distribution sous-jacente. Donc une valeur est aberrante si elle n'appartient pas à cette distribution.

Salvadores *et al.* (2009) présentent une approche basée sur un algorithme nommé LinksB2N pour découvrir les informations chevauchées dans les différentes sources de données RDF en utilisant des méthodes de regroupement (*clustering*). Cette approche a pour but l'automatisation (sans l'intervention humaine) de l'intégration de données qui partagent partiellement le même domaine. LinksB2N cherche les liens implicites entre les graphes par :1) L'identification des prédicats RDF avec des valeurs dispersées où le niveau de dispersion est mesuré grâce à des techniques de clustering de données. 2) La reconnaissance automatique des prédicats RDF qui expriment la même information afin de les comparer ensemble. 3) La comparaison des littéraux RDF en leur associant un facteur de confiance pour chaque comparaison positive.

L'algorithme LinksB2N comporte quatre phases :

1- L'analyse des données à source unique : Consiste à recueillir des statistiques et créer des groupes de valeurs similaires (objets) pour chaque prédicat RDF. Dans cette phase, SPARQL[12] est utilisé pour naviguer entre les graphes.
2- La sélection de prédicats RDF : La sélection se fait en utilisant les groupes créés dans la phase 1 afin de générer des paires de prédicats RDF à comparer.
3- L'évaluation de la valeur du prédicat : Pour chaque paire de prédicats, une évaluation d'équivalence entre les objets RDF est appliquée en attachant un rapport de confiance pour chaque ressource RDF dépendante.
4- La filtration : Cette phase filtre les ressources RDF liées en appliquant une autre itération de la phase 2 pour ceux qui n'ont pas un minimum de facteur de confiance. Cela permet de trouver plus de dépendance entre les instances.

RDF-AI est un outil qui permet l'intégration des ensembles de données RDF par la fusion et l'interconnexion (Scharffe et al.) (2009). Il cherche l'alignement entre deux ensembles de données dans le but de les fusionner et les relier. RDF-AI comprend cinq étapes :

1- Préparation de deux ensembles de données en commençant par la vérification de toute incohérence de jeux de données d'entrée, puis la matérialisation des triplets RDF, ensuite la traduction des propriétés d'une langue à une autre, enfin l'adaptation d'un ensemble de données à l'autre.

2- Application d'une méthode d'alignement entre les deux ensembles de données afin de trouver des propriétés qui se ressemblent.

3- Application d'un module d'interconnexion, en prenant comme entrée l'alignement entre les deux ensembles de données et comme sortie le graphe nommé contenant un ensemble de relations primitives.

---

[12] SPARQL sera présenté dans le chapitre III.

4- Fusion des deux ensembles de données en fonction de l'alignement, les graphes originaux des deux ensembles et l'ensemble des paramètres utilisateur.

5- Vérification des incohérences qui peuvent apparaitre dans les résultats de la fusion.

Nikolov *et al.* (2007) proposent l'approche Knofuss pour fusionner deux différents ensembles de données RDF en intégrant des ontologies au niveau instances. Pour comparer deux ensembles de données, les auteurs proposent d'abord de dédier une ontologie à chaque ensemble en précisant les ressources à comparer. Si les deux ontologies sont différentes, la méthode d'alignement d'ontologie est utilisée. Afin de savoir quelle méthode de similarité utiliser, un contexte d'application pour chaque type de ressource est défini. Ces contextes d'application auront le même identificateur ID dans les deux ontologies de jeux de données s'ils ont la même méthode de similarité. Knofuss vérifie également s'il existe des incohérences dans les résultats de la fusion. Cette approche est composée de deux tâches principales :

1) L'intégration de l'ontologie (responsable de l'alignement du niveau schéma).

2) L'intégration des connaissances (exploitation au niveau instances).

Stumme et Maedche (2001) proposent FCA-MERGE une méthode de fusion des ontologies basée sur AFC, en utilisant une approche de description structurelle du processus de fusion *bottom-up*. Dans cette approche les données sont du niveau instance. Elles sont extraites à partir d'un ensemble de documents texte d'un domaine spécifique en appliquant des techniques de traitement du langage naturel. Une fois les instances extraites l'approche utilise AFC afin de construire le treillis de concepts. L'approche comprend les étapes suivantes :

1- L'extraction des instances et la création des contextes formels : Cette étape consiste à extraire les instances et les utiliser afin de générer pour chaque ontologie un contexte formel.

2- La création du treillis de concepts : Consiste à créer le treillis de concepts à partir des contextes formels générés dans l'étape précédente.

3- La production de l'ontologie finale : Chaque concept formel du treillis de concepts créé dans l'étape précédente est un candidat pour un concept ou une relation dans l'ontologie finale.

David et Scharffe (2012) proposent une approche d'analyse de jeux de données RDF basé sur les dépendances de clefs. Cette approche utilise un algorithme de découverte de clefs (chaque clef identifie de manière unique un tuple d'une relation) adapté aux données RDF. Les clefs découvertes permettent de trouver des liens entre deux jeux de données différents et la détection d'incohérences et erreurs dans le même jeu de données. Cette approche comprend trois étapes :

1- L'application de la méthode *Alignement* des ontologies utilisées par les deux jeux de données.
2- La sélection d'ensemble de propriétés qui permettent de minimiser le nombre de comparaisons nécessaires afin d'identifier les instances en commun.
3- Le choix des mesures de similarité à utiliser afin de comparer chaque paire de propriétés

Aquin et Motta (2011) proposent une approche basée sur l'extraction automatique d'un ensemble de questions auxquelles on peut répondre par un ensemble de données RDF. Cette approche utilise l'analyse formelle de concepts (AFC) pour identifier des ensembles d'objets à partir d'un ensemble de données qui partagent des propriétés communes. Chacun de ces ensembles représente les réponses à une question particulière. L'approche comprend les étapes suivantes :

1- Introduction des notations : Consiste à introduire des notations simples pour décrire un ensemble de donnée RDF en utilisant les instances qui représentent des objets individuels. Par exemple : *Tom* est une instance de la classe *Person*. Ceci peut être représenté par une simple notation *Person(Tom)*.
2- Hypothèses et exigences : Consiste à proposer des questions pour lesquelles les réponses sont des ensembles d'objets où chaque question est caractérisée par un ensemble de propriétés qui sont communes aux éléments de la réponse.
3- Construction des contextes formels : Construire les contextes formels de la forme $C = (G, M, I)$, où G est l'ensemble de toutes les instances de l'ensemble de données. M correspond à toutes les caractéristiques possibles de ces instances.
4- Création du treillis et élimination des redondances : Le treillis de concepts est créé à partir des contextes formels construits dans l'étape précédente.

5- Utilisation de treillis de concept comme une interface de requête : Chaque concept de treillis représente une question, avec son Intention (*Intent*) étant les composantes de la question et l'Extension (*Extent*) les réponses.

De nombreux spécialistes dans le domaine ont proposé des approches et des applications utilisant AFC pour le traitement des données liées, citons quelques-unes. Delteil *et al.* (2002) proposent une approche de construction de treillis de concepts par l'apprentissage des concepts à partir des annotations RDF de documents web. Cette méthode consiste à extraire des descriptions conceptuelles des ressources web du graphe RDF en regroupant toutes les annotations pour ensuite former des concepts à partir de tous les sous-ensembles possibles de ressources. Chekol et Napoli (2013) utilisent le treillis de concepts afin de révéler les sémantiques cachées dans le contenu de la réponse d'une requête SPARQL en ajoutant une couche de concept formel pour le web sémantique. Ceci permet de raffiner les requêtes, de nettoyer les donner et de regrouper (*clustering*) les concepts, etc. Plusieurs d'autres approches ont été discutées par Sertkaya (2011) .

2.3 L'approche que nous proposons:

Notre approche se distingue des approches citées ci-dessus par les points suivants :

- *Création d'une ontologie à partir d'instances* :
  Au départ, dans notre approche, on n'utilise pas du tout les classes de l'ontologie Dbpedia. Les descripteurs issus des instances sont déjà en soi des premières classes pour l'ontologie, puis l'application de l'ARC permet de les organiser dans un schéma de spécialisation/généralisation plus fouillé. Les classes ainsi découvertes forment une ontologie "d'usage". Cette ontologie "d'usage" peut être un meilleur guide ensuite pour créer des objets, mais elle donne en premier lieu une compréhension structurée des concepts d'un domaine.

- *Modèle raffiné de normalisation de modèle de classes* :
  Dans certaines approches telles celles de Hacene (2007) et Miralles *et al.* (2014), le modèle de la FCR (famille de contextes relationnels) est un méta-modèle, inspiré de celui d'UML, dans lequel les entités décrites (contextes formels) sont : classes, attributs, rôles, associations, etc. Les contextes relationnels sont les méta-associations, telles que *hasAttribute* entre classes et attributs. Dans ce méta-modèle, toutes les classes sont dans le même contexte formel. Dans notre cas, nous prévoyons une étape de pre-clustering qui sépare les descripteurs des instances de Dbpedia en plusieurs sous-groupes (comme les descripteurs d'écrivains, les descripteurs de livres, les descripteurs de lieux, etc.). Il y a donc autant de contextes formels des descripteurs que de tels groupes de descripteurs. Cela a une conséquence sur le modèle de la FCR, car les relations sortantes des groupes de descripteurs doivent également être divisées en sous-relations. Cela amène une certaine complication du modèle de la FCR, mais en contrepartie, cela simplifie le processus ARC et évite de créer des généralisations de descripteurs (ou de rôles) sans intérêt.

- *Formulation d'implications* :
  Notre approche permet d'énoncer des implications sur la présence/absence de propriétés. Comme on travaille en itérant sur les relations, on découvre des complétions de données qui n'apparaissent pas sur des données plates.

Tableau 2.1 Tableau comparatif des approches.

	(Fennouh et al., 2015)	(Mehri_Dehnavi, 2014)	(Völker et Niepert, 2011)	(Nikolov *et al.*, 2010)	(Curé *et al.*, 2013)	(Salvadores *et al.*, 2009)	(Scharffe *et al.*, 2009)	(Nikolov *et al.*, 2007)	(Stumme et Maedche, 2001)	(David et Scharffe, 2012)	(d'Aquin et Motta, 2011)	(Delteil *et al.*, 2002)	(Notre approche, 2015)
Permet d'abstraire des descripteurs des classes				X								X	X
Permet d'abstraire des descripteurs des propriétés		X							X	X	X		X
Travaille à partir des données RDF		X	X			X	X	X		X	X	X	X
Travaille à partir d'une ontologie RDFS		X											X
Travaille à partir d'OWL	X												
Exploite les régularités entre propriétés des DL		X				X						X	X
Produit un schéma RDF	X	X	X	X				X			X		X
Exploite les règles d'associations			X										X

CHAPITRE III
MÉTHODOLOGIE ET IMPLÉMENTATION

Ce chapitre présente la méthodologie de la recherche et l'illustre avec des extraits d'instances de Dbpedia (juillet 2015).

3.1 Approche méthodologique

L'approche dans ce mémoire consiste à étudier l'ontologie de Dbpedia dans son usage en pratique, c'est-à -dire au travers de ses instances, et d'en tirer des leçons pour observer les usages et améliorer les pratiques (par exemple modifier la forme de l'ontologie, ou effectuer des recommandations sur la collecte des données). Modifier la forme de l'ontologie pourrait s'appuyer sur des classes souvent observées (mais inexistantes dans l'ontologie d'origine). Effectuer des recommandations pourrait s'appuyer sur l'observation de groupes de propriétés souvent renseignées ensemble.

Dbpedia s'appuie sur une ontologie dont les classes et les propriétés sont disponibles à l'URL suivant : `http://dbpedia.org/ontology/`. Le nombre de propriétés par classe peut être élevé, par exemple la classe Artist (`http://dbpedia.org/ontology/Artist`) dispose par déclaration locale ou par héritage de 281 propriétés, obtenues en comptant le nombre de (is **rdfs:domain** of), dont les instances n'utilisent qu'une partie.

Nous nous intéressons à ces usages par l'extraction du schéma de classes sous-jacent aux instances existantes. Nous nous demandons à quel point la construction de groupes d'instances partageant des groupes de propriétés, détectables par Analyse Relationnelle de Concepts, permet de découvrir ce schéma sous-jacent. A partir de ce schéma sous-jacent, on peut déployer des analyses, comme détecter les propriétés toujours utilisées, toujours utilisées ensemble, jamais utilisées ensemble, ou telles que l'usage de l'une implique l'usage de l'autre, etc.

De cette analyse on pourrait dériver un assistant pour l'extraction qui s'appuie sur les usages précédents et enrichir l'ontologie avec certaines des classes découvertes et un assistant pour la collecte des données (qui indique par exemple lors d'une nouvelle collecte quelles

valeurs de propriétés semblent manquer ou quelles présences de propriétés semblent contradictoires).

Notre approche comporte quatre étapes :

1) L'extraction des données de Dbpedia (niveau instances) et nettoyage des instances pour obtenir des descripteurs. Certaines propriétés seront interprétées comme des attributs (par exemple « *birthdate* »). Les attributs verront leurs valeurs supprimées, on ne gardera que leur nom. D'autres propriétés seront gardées comme des relations (par exemple, le descripteur « *Frankenstein* » garde une relation « *author* » avec le descripteur « *Mary_Shelley* »). Ces relations seront réifiées (comme des rôles) et disposeront de leur propre description. A l'issue de ce nettoyage, on conserve un descripteur pour chaque instance, composé d'attributs et des relations entre les descripteurs. Cette étape est automatique en utilisant les requêtes SPARQL.

2) La transformation des descripteurs en modèle de données pour l'application de l'ARC. Cette application doit permettre de créer des nouveaux descripteurs plus généraux par factorisation d'attributs communs à plusieurs descripteurs (cette partie peut être réalisée avec AFC). Elle va également utiliser des descripteurs créés par factorisation pour créer de nouveaux descripteurs par le biais des relations (cette partie est spécifique à l'ARC). Par exemple, supposons que deux descripteurs d1 et d2 aient été généralisés dans un descripteur d3. Considérons à présent deux descripteurs d3 et d4 avec respectivement un rôle r1 de d3 à d1 et un rôle r2 de d4 à d2. Les rôles r1 et r2 pourront être généralisés comme « rôle aboutissant sur un d1 ou un d2 (c'est-à-dire un d3)», puis les descripteurs d3 et d4 pourront être à leur tour généralisés par un nouveau descripteur d5 introduit pour représenter « le descripteur des instances ayant un rôle aboutissant sur un d3 ». Les descripteurs peuvent être étudiés par groupes ou tous ensemble (ce qui a une influence sur l'étude des rôles). Cette étape se fait manuellement.

3) L'utilisation des résultats de l'ARC pour extraire un modèle de classes (donc en restant au niveau schéma). Les concepts de treillis particuliers (les treillis des descripteurs) sont

utilisés et interprétés comme des classes. Ces concepts contiennent les attributs qui permettent de reconstruire les propriétés (de type *data*). Les concepts des treillis des rôles sont utilisés pour reconstruire des propriétés (de type *object*) entre descripteurs. Dans cette étape, nous faisons appel à l'outil RCAExplore.

4) L'analyse des résultats en comparant avec les classes de Dbpedia et en analysant les distributions d'attributs et de rôles. On étudie aussi la manière dont les données sont renseignées pour proposer des complétions ou mettre en évidence des incohérences.

Dans ce qui suit, chaque étape est décrite en détail et illustrée sur un exemple.

3.1.1 Extraction des descripteurs

A cette étape les ressources et leurs propriétés sont extraites des triplets RDF de la base de données Dbpedia. Notre exemple utilise des données extraites en date de 15/07/2015. Le détail des données est en annexe 1 (extrait des instances) et annexe 2 (descripteurs retenus). Nous avons considéré des entités de type « *Person* » (mais en nous focalisant sur des écrivains), des entités de type « *Work* » en nous focalisant sur des livres, et des entités de type « Place ».

Un descripteur (de niveau schéma) est extrait de chaque instance. Par exemple, soit l'instance *Frankenstein* (`http://dbpedia.org/page/Frankenstein`), qui est décrite par cet extrait dans Dbpedia :

```
dbpedia-owl:abstract (... long text ...)
dbpedia-owl:author dbpedia:Mary_Shelley
dbpprop:language English
dbpprop:genre dbpedia:Gothic_fiction
```

Cette instance va donner naissance au descripteur suivant dans lequel on effectue un choix pour distinguer certaines propriétés comme des attributs (*ici abstract*, *language*, *genre*) et d'autres comme des rôles (*author*). On enlève les valeurs des attributs mais on garde le nom de l'instance pour les rôles, par exemple *Mary_Shelley*. Ce nom d'instance est assimilé au nom du descripteur (de *Mary_Shelley*).

Le descripteur de Frankenstein est ainsi :

```
dbpedia-owl:abstract
dbpedia-owl:author dbpedia:Mary_Shelley
dbpprop:language
dbpprop:genre
```

Ici le choix entre attribut ou rôle a été fait à la main. Une propriété de type simple (date, chaîne de caractères, etc.) est transformée systématiquement en attribut. Une propriété dont la valeur est un URL est transformé en attribut (comme le choix fait pour *genre*) ou en rôle (comme le choix fait pour *author*). Pour une expérimentation plus poussée et automatisée, il conviendra d'utiliser une heuristique ou une liste prédéfinie de ce qui deviendra un attribut ou deviendra un rôle.
Comme indiqué plus haut, les noms des descripteurs et les noms des instances sont confondus pour la suite, le descripteur issu de l'instance *Frankenstein* s'appellera également *Frankenstein.*

3.1.2 Transformation en modèle pour ARC

Pour analyser les données avec l'ARC, les données récoltées à l'étape précédente doivent être transformées dans une FCR composée de contextes formels (entre des objets et des attributs) et de contextes relationnels (entre des objets de contextes formels). Plusieurs choix doivent être effectués à cette étape qui est la plus délicate et sujette à différents paramétrages.

Les descripteurs sont naturellement des objets formels mais ils peuvent être soit tous placés dans un même contexte formel soit séparés en plusieurs contextes formels. Ici nous faisons le choix pour l'exemple de faire un contexte formel par grande catégorie (un pour les écrivains, un pour les livres, un pour les lieux). On s'appuie donc ainsi sur certaines classes de l'ontologie de Dbpedia, que l'on souhaite raffiner. Cela évite de généraliser un descripteur de personne et un descripteur de lieu, mais il existe peut-être des cas où une telle généralisation serait intéressante.

Les descripteurs sont décrits par des attributs formels qui sont les attributs que nous avons spécifiés précédemment (choisis parmi les propriétés de Dbpedia).

Le contexte formel des descripteurs des instances de personnes est défini (celui des travaux et celui des lieux sont en annexe 3) dans le tableau 3.1.

Tableau 3.1 Contexte formel des descripteurs des instances de Personnes.

	name	birthDate	deathDate	activeYearsStartYear	genre	awards	pseudonym	website	nationality
EdgarPoe	X	X	X						
MaryShelley	X	X	X						
StephenKing	X	X		X	X	X	X	X	X
BrianKeene	X	X		X	X			X	X

Les rôles sont réifiés afin de pouvoir les organiser entre eux par groupes de rôles partageant des aspects communs (leur nom, aspect qui sera connu par un attribut formel, mais également des extrémités, origine ou destination, communes qui seront connues par l'intermédiaire des contextes relationnels). Réifier un rôle signifie que chaque propriété

considérée comme un rôle devient un objet formel. Pour distinguer deux rôles correspondant à deux propriétés portant le même nom, on les préfixe par le nom du descripteur dont ils proviennent, par exemple on appellera « *EdgarPoe::birthPlace* » le rôle *birthplace* du descripteur *EdgarPoe* et « *MaryShelley::birthPlace* » le rôle *birthplace* du descripteur *MaryShelley*. Distinguer ces rôles permettra qu'ils portent ou non le même nom, de les grouper suivant d'autres aspects comme leurs origines ou leurs extrémités, comme on le verra par la suite avec les contextes relationnels.

Dans un premier temps, on crée donc un contexte formel par ensemble de rôles entre deux ensembles d'objets de deux contextes formels connectés par des propriétés. Plus précisément, on aura un tel contexte formel pour chaque paire d'ensembles de descripteurs connectés. Par exemple, les rôles entre livres (travaux) et personnes donnent lieu à un premier contexte formel, un peu particulier car tous les rôles ont le même nom comme illustré dans le tableau 3.2.

Tableau 3.2 Contexte formel RoleBookWriter.

	author
Frankenstein::author	X
ArthurGordonPym::author	X
SixStories::author	X
DeadSea::author	X

Les rôles entre personnes et lieux sont décrits dans un autre contexte formel. Dans ce dernier, deux noms de rôles sont possibles et on a également choisi comme caractéristique la chaîne « place » qui est un suffixe significatif des deux autres (voir tableau 3.3).

Tableau 3.3 Contexte formel RoleWriterPlace.

	birthPlace	deathPlace	place
EdgarPoe::birthPlace	X		X
MaryShelley::birthPlace	X		X
StephenKing::birthPlace	X		X
BrianKeene::birthPlace	X		X
EdgarPoe::deathPlace		X	X
MaryShelley::deathPlace		X	X

S'il y avait également un ensemble de rôles entre *Book* et *Place*, on trouverait un autre contexte formel nommé «*RoleBookPlace* » pour les décrire (ils ne seraient a priori pas mélangés avec les rôles entre travaux et écrivains : c'est également un choix modifiable).

L'apport de l'ARC va se révéler pleinement grâce au codage des relations entre les objets des différents contextes. Il y a là encore différents choix possibles. Les informations essentielles qui sont codées dans les contextes relationnels sont les suivantes :

- Le fait que les descripteurs possèdent des rôles : ce sera codé par un contexte relationnel pour chaque paire (contexte formel de descripteurs, contexte formel de rôles).
- Le fait que les rôles aient des descripteurs décrivant le type de leur origine et le type de leur destination.

Un exemple de contexte relationnel pour les descripteurs de travaux et les rôles que peuvent posséder ces travaux serait celui du tableau 3.4.

Tableau 3.4 Contexte relationnel hasRoleBookWriter.

	Frankenstein::author	ArthurGordonPym::author	SixStories::author	DeadSea::author
Frankenstein	X			
ArthurGordonPym		X		
SixStories			X	
DeadSea				X

La forme simple de ce contexte (diagonale de croix) ne doit pas tromper sur son importance : c'est par ce contexte relationnel que va passer le raffinement des groupes de livres suivant les groupes d'écrivains observés.

Un exemple de contexte relationnel décrivant quels descripteurs sont à l'origine des rôles entre travaux et personnes (le contexte dual *hasDestinationBookWriter* sera également présent) est présenté dans le tableau 3.5.

Tableau 3.5 Contexte relationnel has OriginBookWriter.

	Frankenstein	ArthurGordonPym	SixStories	DeadSea
Frankenstein::author	X			
ArthurGordonPym::author		X		
SixStories::author			X	
DeadSea::author				X

La famille de contextes complète est présentée en annexe 3. Remarquons que dans notre approche, les propriétés-attributs n'ont pas été réifiées mais une variante consisterait à le faire pour généraliser par exemple à partir de leurs types, de parties des noms d'attributs ou d'autres hyperonymes.

Pour passer au cas général, nous utilisons donc une famille de contextes relationnels composée des éléments suivants :

- Un contexte formel de descripteurs par ensemble de descripteurs choisi ; dans ces contextes, les attributs formels sont les propriétés-attributs choisies.

- Un contexte formel de rôles par paire d'ensembles de descripteurs (entre lesquels il y a des rôles); dans ces contextes les attributs formels sont les noms des rôles.

- Un contexte relationnel « hasRoleDR » par paire (D, R) où D est un ensemble de descripteurs et R un ensemble de rôles allant de D vers Df un autre ensemble de descripteurs.

- Un contexte relationnel « hasOriginRD » par paire (R, D) où R est un ensemble de rôles et D un ensemble de descripteurs origine de ce rôle.

- Un contexte relationnel « hasDestinationRD » par paire (R, D) où R est un ensemble de rôles et D un ensemble de descripteurs destination de ce rôle.

3.1.3 L'utilisation des résultats de l'ARC

Une fois la famille de contextes relationnels définie, on applique un outil de construction des treillis, comme RCAExplore (`http://dolques.free.fr/rcaexplore/`) ou Galicia (`http://www.iro.umontreal.ca/~galicia/`) . Lors de cette application, on choisit, comme dans la reconstruction de modèles UML, l'opérateur de Scaling existentiel.

En appliquant RCAExplore, on obtient l'ensemble des treillis de chaque étape de la construction.

- **Treillis des concepts formels *écrivains***

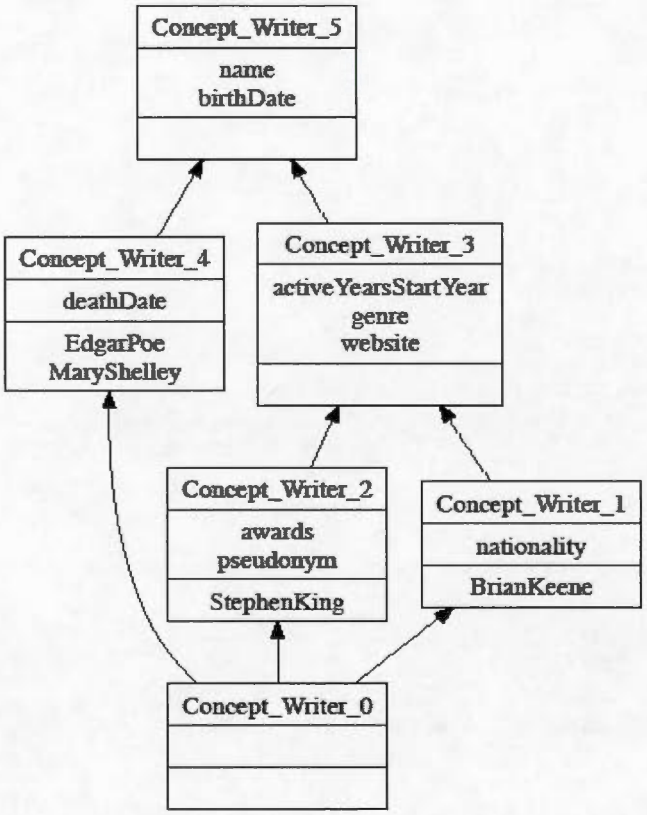

Figure 3.1 Treillis des concepts formels Writers à l'étape initiale.

La figure 3.1 représente le treillis des concepts des écrivains à l'étape initiale.

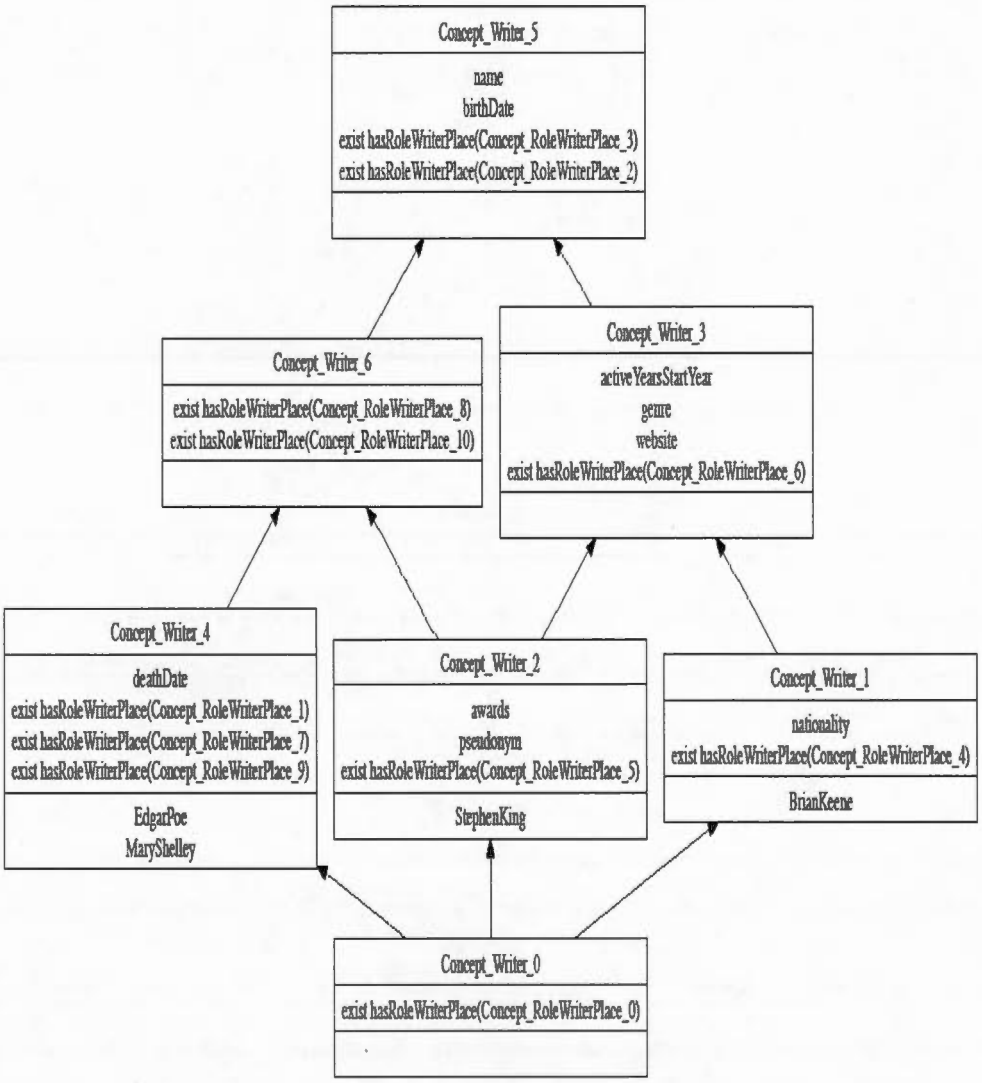

Figure 3.2 Treillis des concepts formels Writers à l'étape finale.

La figure 3.2 représente le treillis des concepts des écrivains à l'étape finale.

- **Treillis des concepts formels *livres***

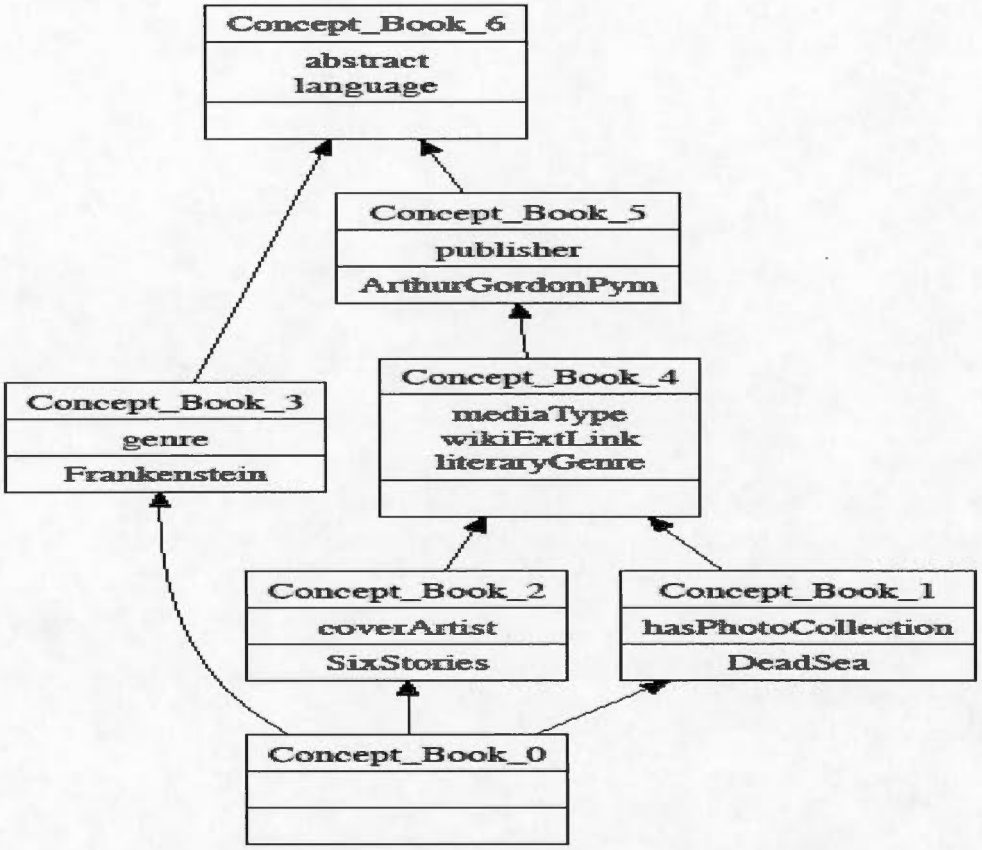

Figure 3.3 Treillis des concepts formels Books à l'étape initiale.

La figure 3.3 représente le treillis des concepts formels des livres à l'étape initiale.

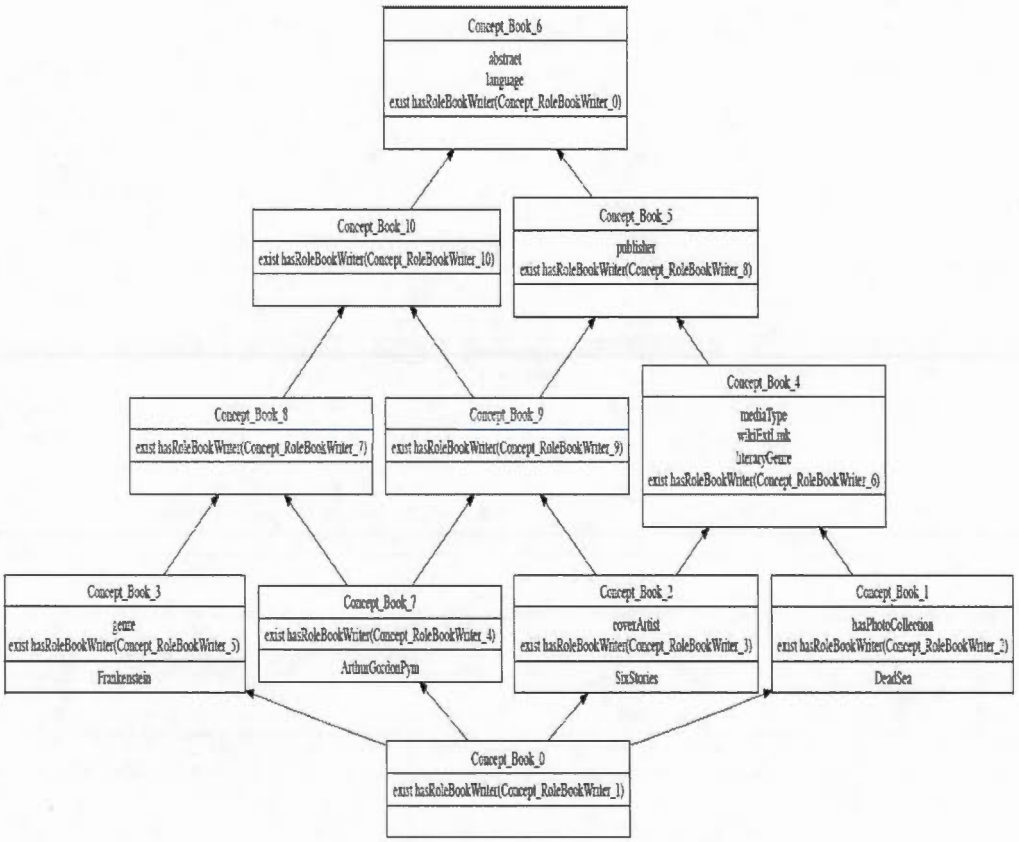

Figure 3.4 Treillis des concepts formels Books à l'étape finale.

La figure 3.4 représente le treillis des concepts formels des livres à l'étape finale.

- **Treillis des concepts formels *lieux***

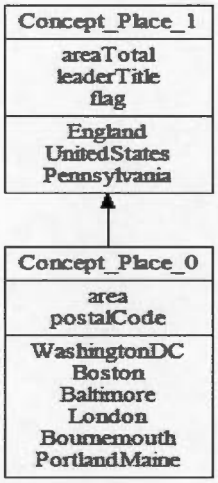

Figure 3.5 Treillis des concepts formels places (lieux).

La figure 3.5 représente le treillis des concepts formel des lieux à l'étape initiale et aussi l'étape finale (pas de changement).

- **Treillis des concepts formels *RoleBookWriter***

Dans l'étape initiale de la construction des treillis, il existe un seul concept formel *Concept_RoleBookWriter*.

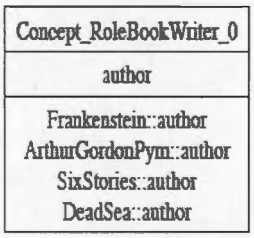

Figure 3.6 Treillis des concepts formels RoleBookWriter à l'étape initiale.

La figure 3.6 représente le treillis des concepts formels *RoleBookWriter* qui montre les rôles entre les livres et les écrivains à l'étape initiale.

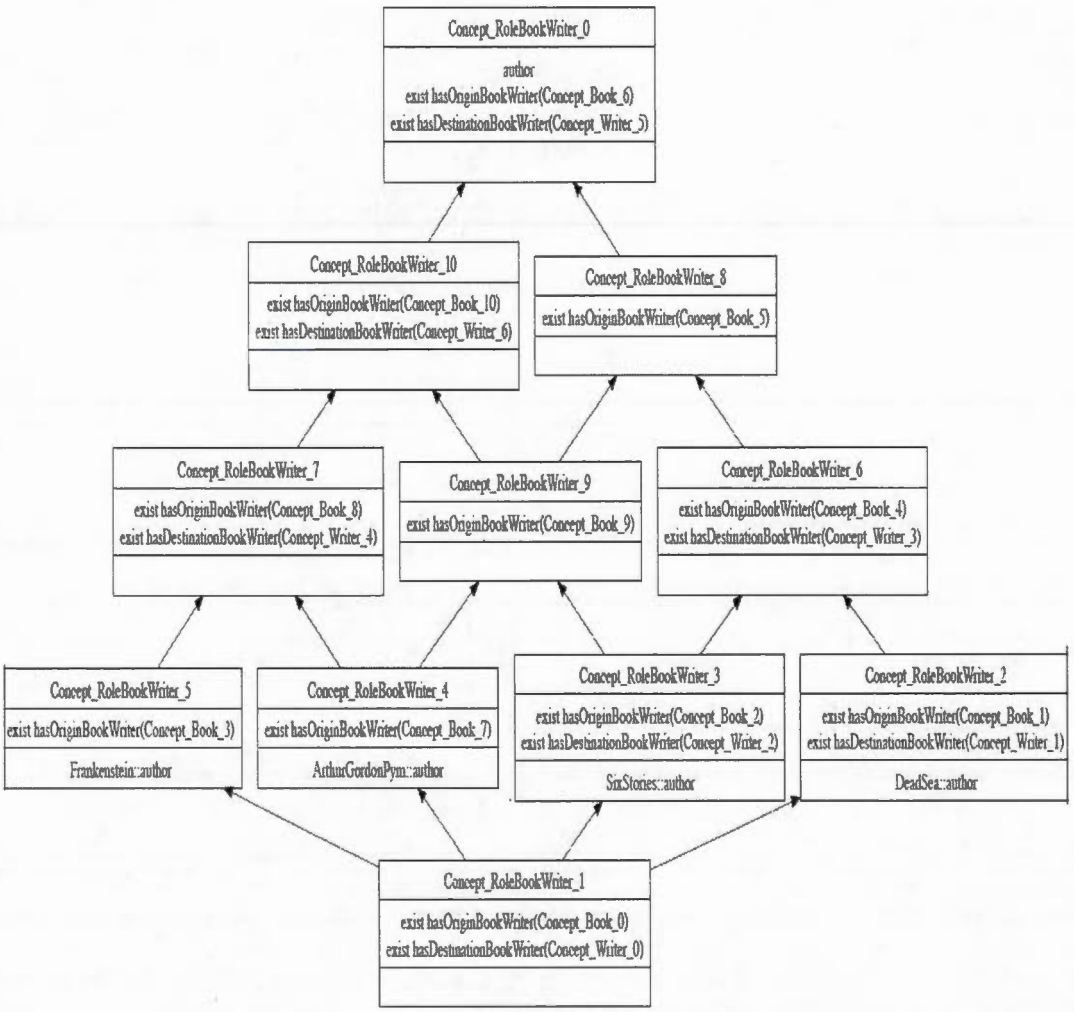

Figure 3.7 Treillis des concepts formels RoleBookWriter à l'étape finale.

La figure 3.7 représente le treillis des concepts formels *RoleBookWriter* qui montre les rôles entre les livres et les écrivains à l'étape finale.

- **Treillis des concepts formels RoleWriterPlace**

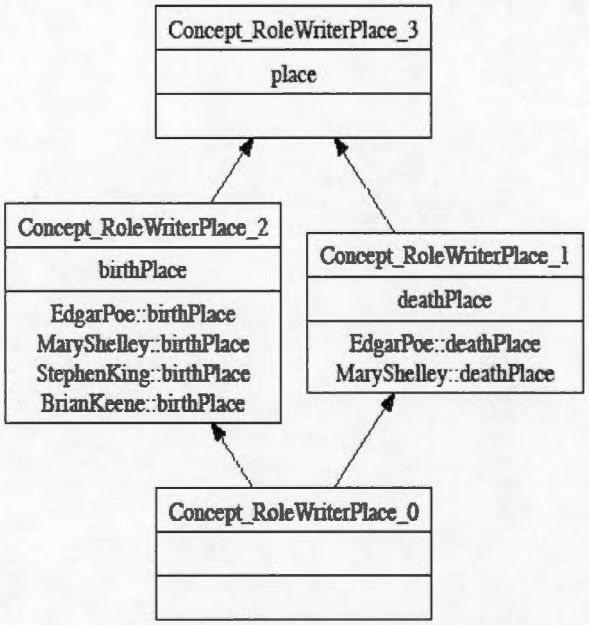

Figure 3.8 Treillis des concepts formels RoleWriterPlace à l'étape initiale.

La figure 3.8 représente le treillis des concepts formels *RoleWriterPlace* qui montrent les rôles entre les écrivains et les lieux à l'étape initiale.

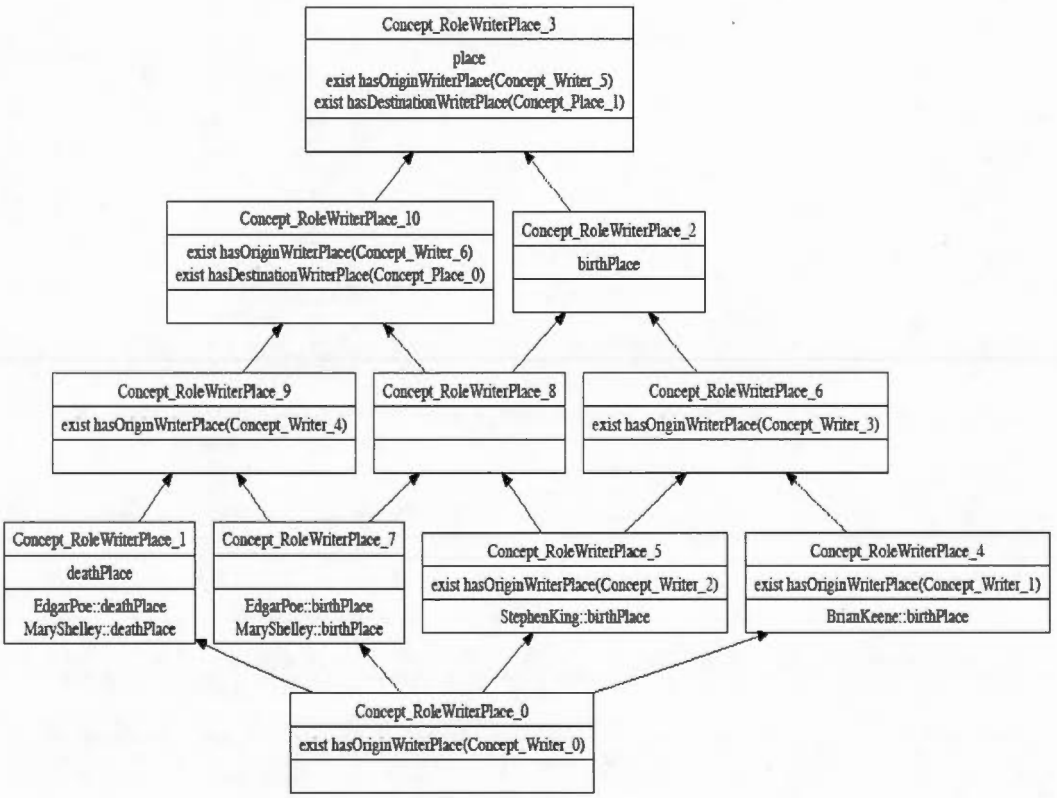

Figure 3.9 Treillis des concepts formels RoleWriterPlace à l'étape finale.

La figure 3.9 représente le treillis des concepts formels *RoleWriterPlace* qui montrent les rôles entre les écrivains et les lieux à l'étape finale.

3.1.4 Analyse des résultats

Pour mieux analyser les résultats obtenus, nous allons réduire les noms des concepts, comme par exemple : *Concept_RoleWriterPlace_6* devient *CRWP_6*, *Concept_Writer_6* devient *CW_6*. On réduit aussi les noms de rôles comme : *exist hasRoleBookWriter* devient *∃hRBW*, *exist hasRoleWriterPlace* devient *∃hRWP*. Pour faciliter la compréhension, nous allons présenter une interprétation des données et des concepts créés grâce à la notation UML.

3.1.4.1 Analyse de l'étape initiale de l'ARC

L'étape initiale du processus nous permet d'observer l'effet qu'a une approche basée uniquement sur AFC.

a- Analyse du treillis des lieux

Dans le treillis des lieux (figure 3.5), on trouve deux concepts, le concept *CP_1* correspond au groupe des objets *England*, *UnitedStates* et *Pennsylvania* qui sont décrits par les trois attributs *areaTotal*, *leaderTitle* et *flag*, tandis que le concept *CP_0* correspond au groupe des objets *WashingtonDC*, *Boston*, *Baltimore*, *London*, *Bournemouth* et *PortlandMaine* qui sont décrits par les deux attributs *area* et *postalCode* donnant des informations plus précises sur lieux (qui sont dans ce cas des villes).

b- Analyse du treillis des écrivains

Dans le treillis des écrivains (figure 3.1), les concepts *CW_2* et *CW_1* correspondent aux objets *StephenKing* et *BrianKeene*. Le concept *CW_4* regroupe les descripteurs *EdgarPoe* et *MaryShelley* en factorisant leur propriété commune *deathDate* et donc il correspond à la classe des écrivains décédés. Le concept *CW_3* factorise des propriétés communes aux objets *StephenKing* et *BrianKeene*. On peut l'interpréter comme la classe des écrivains contemporains (ils n'ont pas *deathDate*) et entre autres attributs, ils ont un site web, que l'on peut voir comme un signe de modernité. Le concept *CW_5* représente la classe de tous les écrivains, elle factorise les deux propriétés universellement présentes *name* et *birthDate*.

c- Analyse du treillis des livres

Dans le treillis des livres (figure 3.3), les concepts CB_1, CB_2, CB_3 et CB_5 correspondent aux objets des livres *DeadSea*, *SixStories*, *Frankenstein* et *ArthurGordonPym*, respectivement. Le concept CB_4 regroupe les descripteurs *DeadSea* et *SixStories* en factorisant leurs propriétés communes *mediaType*, *wikiExtLink* et *literaryGenre*. Le concept CB_6 représente la classe de tous les livres, lesquels partagent les deux propriétés *abstract* et *language*.

d- Analyse des treillis de rôles

Dans le treillis initial des rôles entre les écrivains et les lieux (figure 3.8), on observe le concept *CRWP_3* qui groupe tous les rôles (attribut commun « place »). Le concept *CRWP_1* groupe tous les rôles « *deathPlace* » et le concept *CRWP_2* groupe tous les rôles

« *birthPlace* ». Dans le treillis des rôles entre livres et écrivains (figure 3.6), on note un unique concept groupant tous les rôles d'« *author* ».

3.1.4.2 Analyse des résultats de l'ARC et complétion de schéma

Dans cette partie nous nous concentrons sur l'analyse des treillis des livres et des écrivains. Lors de l'analyse, il sera nécessaire de naviguer sur les autres treillis pour expliquer la construction.

a- Analyse du treillis des écrivains

Comparé au treillis des écrivains de l'étape initiale, le treillis de l'étape finale (figure 3.2) présente un concept supplémentaire[13] et des attributs relationnels distribués sur tout le treillis.

Le nouveau concept ici est le concept *CW_6*, qui regroupe *EgdarPoe*, *MaryShelley* et *StephenKing*. Etudions sa genèse. Nous avons vu précédemment que le concept de lieux *CP_0* groupait les villes (lieux décrits par un code postal notamment). Le treillis des rôles entre écrivains et lieux (figure 3.9) fait apparaître un groupe de rôles, dans l'extension du concept *CRWP_10*, qui ont pour destination *CP_0* (ce qui est représenté par l'attribut relationnel *∃hDWP(CP_0)*). La relation entre le nouveau concept *CW_6* et le concept *CP_0* est représentée dans la figure 3.10.

---

[13] Habituellement, un grand nombre de concepts nouveaux est construits par l'ARC. Ici on a voulu limiter le nombre pour des raisons de lisibilité.

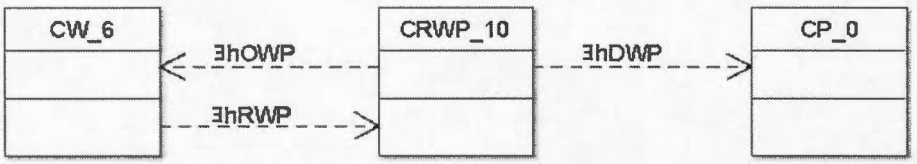

Figure 3.10 Relations entre les concepts : *CW_6* et *CP_0*.

Les trois écrivains *EgdarPoe*, *MaryShelley* et *StephenKing* partagent le fait de posséder un des rôles de l'extension du concept *CRWP_10:EdgarPoe* possède par exemple *EdgarPoe::birthPlace*, *MaryShelley* possède *MaryShelley::birthPlace* et *StephenKing* possède *StephenKing::birthPlace*. Cette caractéristique commune les amène à posséder l'attribut relationnel *∃hRWP(CRWP_10)* qui est factorisée par le concept *CW_6*.

Autre observation, si on examine le concept *CRWP_10*, il groupe également les rôles qui ont pour origine *CW_6* (attribut relationnel *∃hOWP(CW_6)*. Cette information permet de reconstruire une association UML entre les classes qui vont représenter les écrivains « dont les lieux de naissance et de décès sont connus avec précision » et la classe des lieux de ce type.

b- Analyse du treillis des livres

Dans le treillis final des livres (figure 3.4), quatre nouveaux concepts apparaissent, ainsi que des attributs relationnels.

Le concept *CB_7* (livre *ArthurGordonPym*) vient de l'attribut relationnel *∃hRBW(CRBW_4)*; *CRBW_4*, de son côté, vient de l'attribut relationnel *∃hOBW(CB_7)*. Ce concept existe donc par le fait que *ArthurGordonPym::author* a une propriété unique (qu'il est le seul à avoir) qui est de sortir du livre *ArthurGordonPym*. Un tel concept n'est pas une véritable découverte de classe.

Le concept *CB_8* (qui groupe *Frankenstein* et *ArthurGordonPym*) vient de l'attribut relationnel *∃hRBW(CRBW_7)*. *CRBW_7* vient de la factorisation de *∃hRBW(CW_4)*, donc représente les rôles aboutissant sur *CW_4* (écrivains décédés).

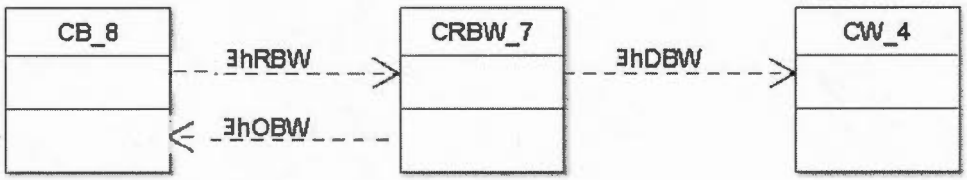

Figure 3.11 Relations entre les concepts : *CB_8* et *CW_4.*

Le concept *CB_10* (groupe *Frankenstein*, *ArthurGordonPym* et *SixStories*) vient de l'attribut relationnel *∃hRBW(CRBW_10)*. Ce dernier concept *CRBW_10* groupe les rôles ayant *∃hDBW(CW_6)*, c'est-à-dire une destination vers un écrivain « dont le lieu de naissance ou de décès est connu avec précision ».

Le concept *CB_9* (qui groupe *ArthurGordonPym* et *SixStories*) vient de l'attribut relationnel *∃hRBW(CRBW_9)*. *CRBW_9* vient de la factorisation de *∃hOBWr(CB_9)* donc représente les rôles aboutissant sur les écrivains qui ont un *publisher* et « dont le lieu de naissance ou de décès est connu avec précision ».

Le concept *CB_4* a l'attribut relationnel *∃hRBW(CRBW_6)* ; *CRBW_6* a *∃hDBW(CW_3)*. *CB_4* est la classe des livres écrits par des écrivains contemporains et on peut observer qu'ils disposent d'une description plus moderne (*mediaType*, *wikiExtLink*) que ne le sont les livres de la classe des livres écrits par des écrivains décédés.

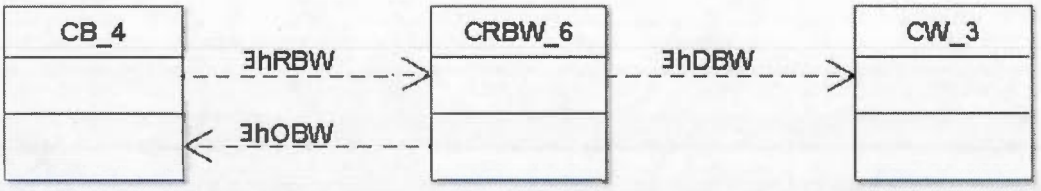

Figure 3.12 Relation entre les concepts : *CB_4* et *CW_3.*

3.1.4.3 Complétion des données

Dans le treillis des écrivains (figure 3.1) on peut (en s'appuyant sur la notion de règle d'implication) observer des régularités sur le renseignement des données : *genre* → *website*. Quand on a une valeur pour l'attribut *genre*, on en a toujours une également pour l'attribut *website* (et réciproquement).

Grâce à l'ARC on peut avoir des informations sur la manière dont les données sont renseignées, mais de manière plus complète qu'avec AFC, car cette fois on en a en traversant les rôles. Par exemple, grâce aux concepts décrits ci-dessus (*CB_4, CRBW_6, CW_3)* on peut observer la règle d'implication : *mediaType* → *website* qui indique que lorsque les livres ont un *mediaType* alors les écrivains correspondant ont un *website*.

Une autre règle d'implication : *nationality* → *∃hRWP (CRWP_6)* car tout objet ayant *nationality* a l'autre attribut. Cette règle d'implication a un support de 1/4 (25%) car une seule instance sur quatre a *nationality* et une confiance de 1=1/1 (100%) car une seule instance a *nationality* et *∃hRWP (CRWP_6)* sur une seule qui a *nationality*.

Prenons la règle d'association suivante : *birthDate* → *∃hRWP (CRWP_10)* avec :

- Un support de 3/4 (75%) car trois instances ont *birthDate* et *∃hRWP(CRWP_10)* sur les quatre instances étudiées.
- Une confiance de 3/4 car trois instances ont *birthDate* et *∃hRWP(CRWP_10)* sur les quatre qui ont *birthDate (*c'est-à-dire 75% des objets qui ont un *birthDate* ont aussi un attribut qui indique un lieu de naissance précis (ville, village)).

La seule exception est *BrianKeene* qui n'a pas un lieu de naissance précis, car le rôle *BrianKeene::birthPlace* aboutit, lui, sur CP_1. En effet la propriété *BrianKeene::birthPlace* pointe sur la Pennsylvanie, qui n'est pas une ville, contrairement aux autres écrivains.

Cela met en évidence une sorte d'incohérence dans la manière dont les données sont renseignées, et par ailleurs *BrianKeene* étant un écrivain contemporain, il est curieux que l'on

ne connaisse pas sa ville de naissance précise. Une recherche sur Wikipedia[14] a confirmé que son lieu de naissance n'est pas indiqué (l'auteur ayant passé une partie de son enfance en Pennsylvanie).

Un mécanisme pour l'aide à la gestion de données liées

Donc, avec les règles d'associations dont la confiance est assez élevée sur des gros jeux de données, on va pouvoir repérer des instances qui ne vérifient pas ces règles et par conséquent dont on peut penser qu'elles sont mal renseignées, ce qui donne une évaluation de la qualité des données existantes. En se basant sur cette évaluation, on peut proposer une complétion de ces données ou une amélioration de sa qualité. Par exemple, reprenons le cas de *BrianKeene,* qui n'a pas un lieu de naissance précis. Nous pouvons proposer une recherche dans les archives papiers afin de trouver sa ville ou son village de naissance.

Un mécanisme pour l'aide à la saisie des données liées

Avec les règles d'implication, on va pouvoir créer un mécanisme de suggestion en exploitant les régularités sur le renseignement des données. Ce mécanisme aide à concevoir un modèle de saisie de données en s'appuyant sur les attributs qui engagent d'autres attributs.

Exemple : la règle d'implication *genre* → *website* ne s'applique pas dans le cas de l'écrivain *Paulo Coelho* et son livre *Aleph* car on trouve la propriété *genre* qui définit son livre mais l'écrivain ne possède pas de site web.

Le treillis construit à l'aide de RCA permet de supporter un bon nombre d'opérations de jeux de données liées. D'une part la connaissance exprimée par le treillis de concepts peut être utilisée pour enrichir le schéma dont les données sont issues et même d'en créer une lorsque un tel n'existe pas. En effet la traduction est directe : un concept formel sur les entités devient une classe OWL définie, dont la définition provient de l'intent du concept. De façon similaire, les concepts sur les rôles peuvent être combinés à la définition des propriétés RDFS. Dans les

---

[14] https://en.wikipedia.org/wiki/Brian_Keene. Consulté le 20/08/2015

deux cas, les liens *sous-concept-de* se traduisent par des propriétés *rdfs :subclassOf* et *rdfs :subproperOf*, respectivement.

Modèle UML représentant les classes

Le modèle UML de la figure 3.13 représente les classes issues des concepts des trois treillis : écrivains, livres et lieux. Ce modèle nous permet de montrer les classes extraites et les relations de spécialisations entre ces classes et entre les relations. Par exemple, la classe *CW_5* représente tous les écrivains, donc tous les écrivains partagent les attributs *name* et *birthDate*. La classe *CB_6* représente tous les livres qui partagent les attributs *abstract* et *language*. La classe *CP_1* représente tous les lieux qui partagent *areaTotal* et *leaderTitle*. Dans ce modele on peut constater plusieurs spécialisations entre les classes telle que celle entre la classe *CW_5* et les deux classes *CW_6, CW_3* qui distinguent deux catégories d'écrivains (Contemporain et Décédé). Une autre spécialisation entre *CP_1* et *CP_0* est celle distinguant les villes des pays. Le diagramme montre aussi les spécialisations entre les relations telle que celle entre la relation *CRWP_10* et *CRWP_3*.

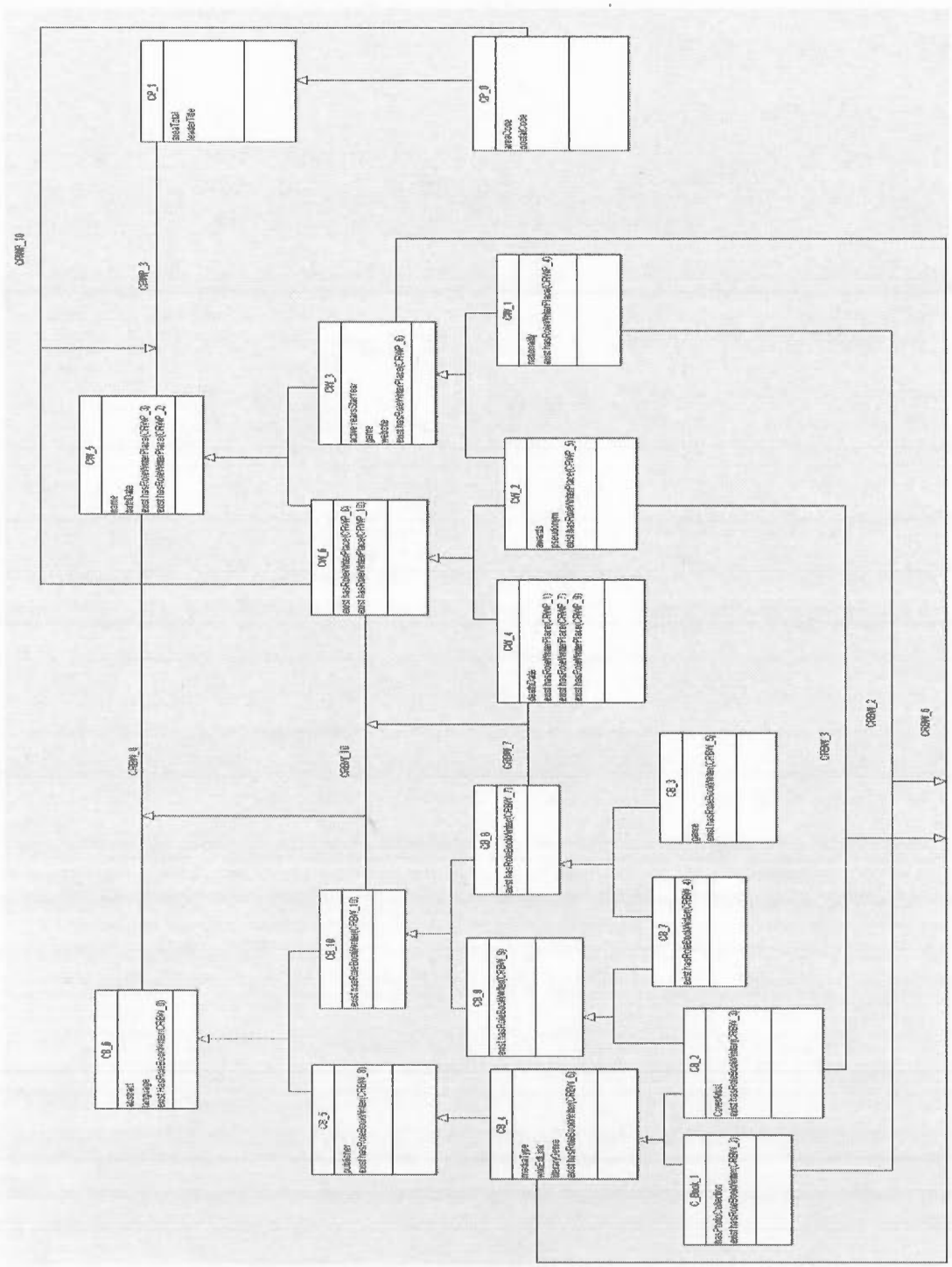

Figure 3.13 Modèle UML.

3.2 Outils utilisés

Cette section présente les outils et les plateformes utilisés afin de mettre en œuvre notre approche.

3.2.1 JENA[15]

Nous avons utilisé Jena pour extraire les propriétés et les ressources de données RDF. Jena a été développé par les membres du groupe de recherche du web sémantique dans les laboratoires HP. Jena est conforme aux spécifications RDF afin de faciliter la création et la manipulation de ses graphes. Jena a été construit en premier temps uniquement pour RDF, puis des nouvelles fonctionnalités ont été ajoutées pour l'utilisation du langage OWL (McBride) (2001).

3.2.2 SPARQL

Le moteur de recherche ARQ pour Jena permet l'intégration des données accessibles stockées dans un modèle en fournissant un langage de requêtes appelé SPARQL. Ce langage de requêtes standard est conçu pour être utilisé dans le modèle de type RDF. SPARQL interroge les informations de la même manière que SQL fait mais il utilise le modèle RDF au lieu des modèles de données relationnelles. Il utilise du *pattern matching* sur les données en graphe. SPARQL offre plusieurs types de solutions, selon la forme de requêtes telles que:

- *Requête Select* : Renvoie des tuples où chaque composante d'un tuple correspond à la valuation d'une variable de la requête par un terme RDF.
- *Requête Construct* : Renvoie un graphe RDF utilisant les évaluations des variables présentes dans la requête.
- *Requête Ask* : Renvoie un booléen en fonction de la satisfiabilité de la requête.

---

[15] https://jena.apache.org/. Consulté le 20/08/2015

Les requêtes SPARQL sont envoyées dans VIRTUOSO Universal Server afin d'accéder aux données RDF. La figure 3.14 montre l'éditeur des requêtes SPARQL auquel on peut accéder par l'URL `http://dbpedia.org/sparql`.

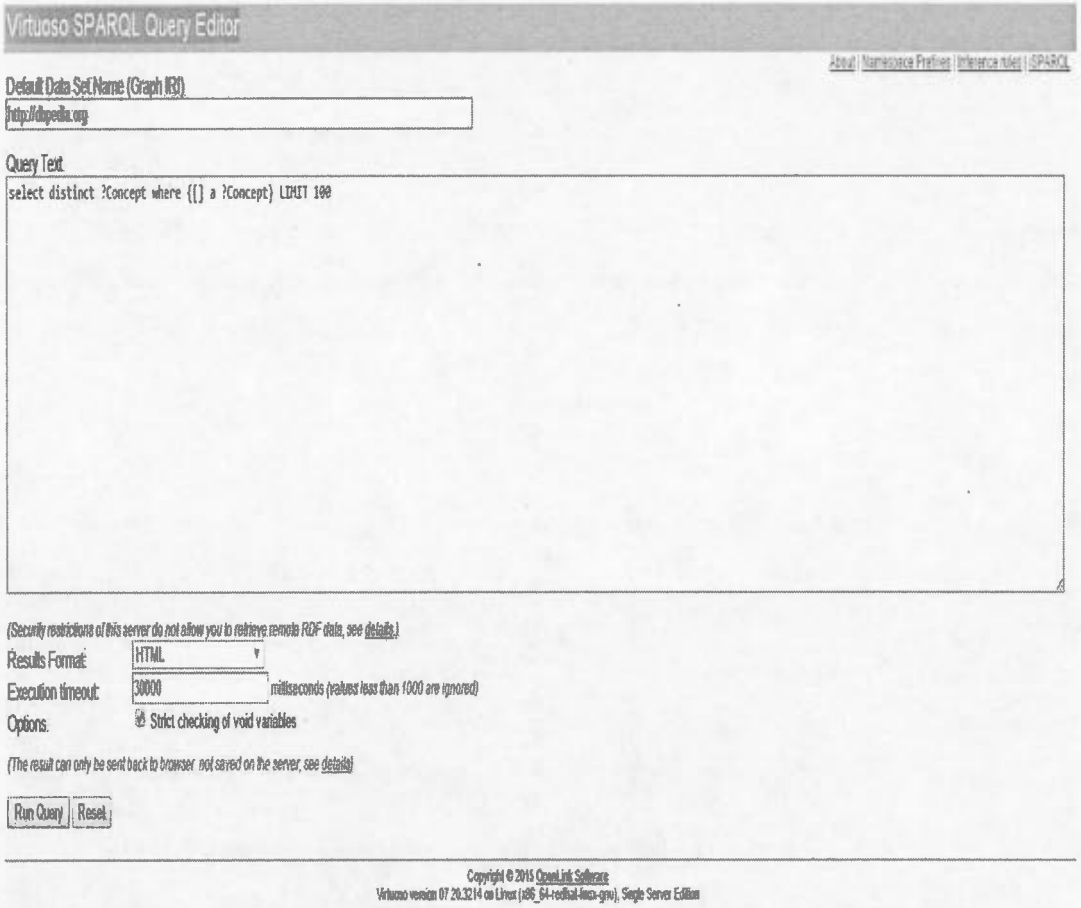

Figure 3.14 Éditeur des requêtes SPARQL.

3.2.3 RCAExplore[16]

Nous avons utilisé RCAExplore afin de générer les treillis de concepts formels et relationnels. RCAExplore est un outil développé en Java par Xavier Dolques et Jean-Rémy Falleri en se basant sur l'algorithme AOC-POSET implémenté par Alain Guiterrez. RCAExplore offre des nouvelles façons de traiter les relations en proposant un éditeur de famille de contextes relationnels, un générateur interactif et un navigateur de treillis de concepts. Il permet l'analyse des concepts relationnels en autorisant le Scaling des contextes exploitant les différents opérateurs tels que : *EXIST, FORALL, CONTAIN, etc.*

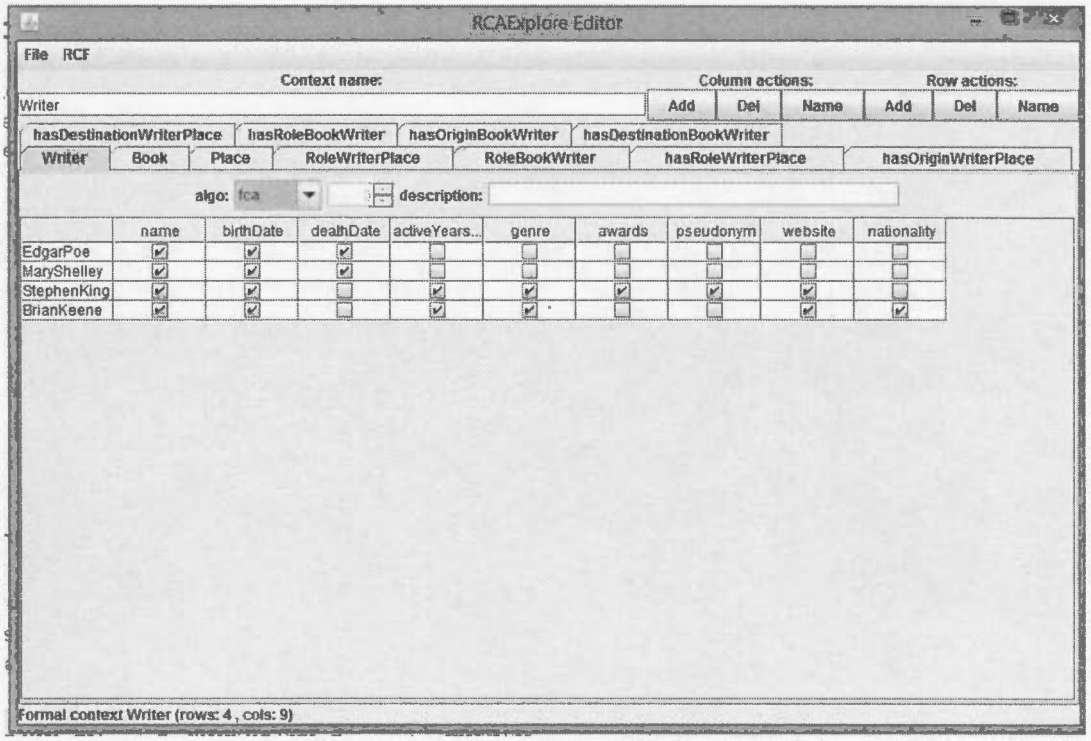

Figure 3.15 RCAExplore.

[16] http://dolques.free.fr/rcaexplore/. Consulté le 20/08/2015

3.2.4 GRAPHVIZ[17]

Graphviz est un logiciel de visualisation graphique open source. Cet outil représente l'information structurelle sous forme de diagrammes abstraits et de réseaux. Il a d'importantes applications dans les réseaux, la bio-informatique, le génie logiciel, les bases de données et la conception de sites Web, l'apprentissage machine, et dans les interfaces visuelles pour d'autres domaines techniques. Grâce aux algorithmes utilisés par Graphviz, cet outil est capable de représenter des graphes denses avec un grand nombre de nœuds.

Nous avons utilisé Graphviz afin de visualiser les fichiers générés par RCAExplore. Ce choix est basé sur la possibilité de personnaliser le rendu des graphes en choisissant les formes, les couleurs et les polices de caractères. De plus Graphviz offre un format des fichiers d'entrée simple et des formats de sortie variés.

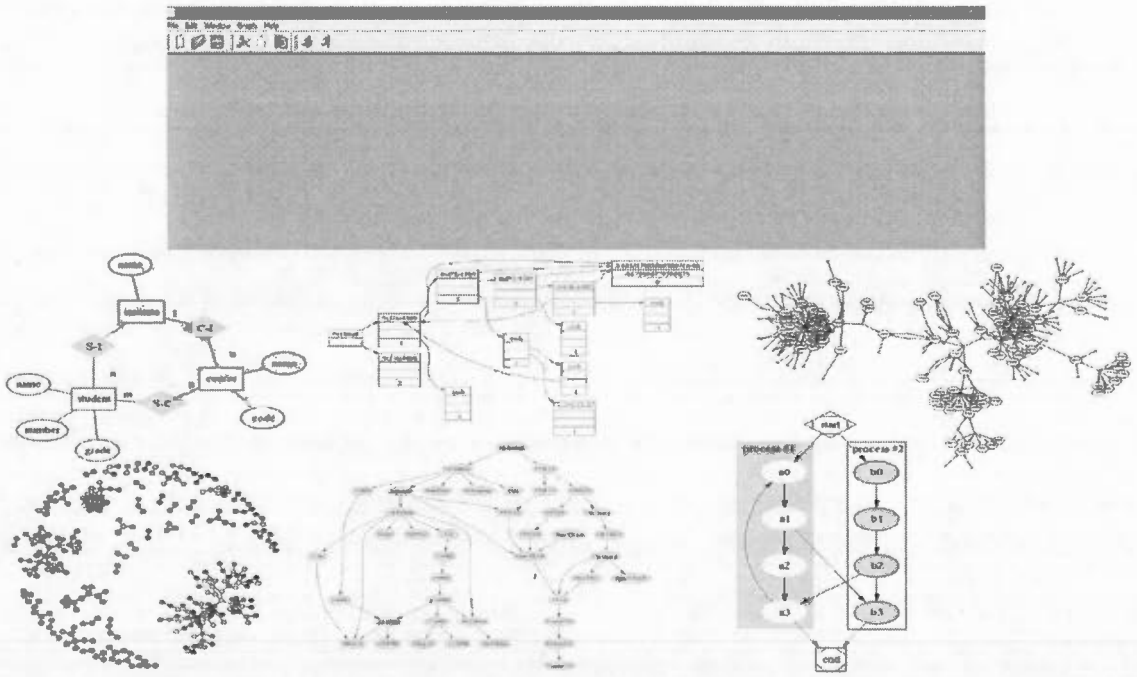

Figure 3.16 GRAPHVIZ.

[17] http://www.graphviz.org/Home.php

CONCLUSION

Dans ce mémoire, nous avons montré comment l'Analyse Relationnelles de Concepts permettait d'extraire un schéma ontologie plus précis pour Dbpedia. Ce schéma correspond à l'usage fait en pratique des classes et des propriétés de l'ontologie initiale. Par la même construction, on peut découvrir des règles sur la distribution des propriétés entre les instances qui permettent de vérifier la cohérence des informations et de les compléter ou de les corriger le cas échéant.

Nous avons présenté une approche basée sur l'Analyse Relationnelle de Concepts. Elle consiste à étudier l'ontologie de Dbpedia en parcourant ses instances afin d'observer les usages et améliorer les pratiques, comme la détection des erreurs et des incohérences dans les données. Cela peut nous aider à proposer des recommandations afin de corriger et compléter l'ontologie. Notre approche comporte quatre phases : 1) L'extraction des données de Dbpedia (niveau instances). 2) La transformation des descripteurs en modèle de données pour l'application de l'ARC. 3) L'utilisation des résultats de l'ARC pour extraire un modèle de classes. 4) L'analyse des résultats en comparant avec les classes de Dbpedia.

Ce travail montre aussi comment l'Analyse Relationnelle de Concept (ARC) peut être plus adéquate que l'Analyse Formelle de Concepts AFC (approche qu'elle étend) dans cette opération de découverte de schéma et de règles. En effet, l'ARC traverse les propriétés à une profondeur quelconque et ramène des informations au travers de ces relations en offrant des mécanismes pour la construction du jeu de données liées et son schéma.

Ce travail contribue à offrir un assistant pour l'extraction de données qui s'appuie sur les usages précédents et à enrichir l'ontologie avec certaines des classes découvertes et un assistant pour la collecte des données en indiquant les valeurs de propriétés manquantes.

Le manque de temps et l'absence d'outils plus avancées capables de trier les concepts non pertinents nous ont empêché de faire une étude plus substantielle, car la taille des treillis risquait de s'avérer un facteur prohibitif pour tout examen détaillé des concepts formels ainsi que des règles.

Les perspectives futures de notre travail sont de plusieurs ordres. Une expérimentation en grandeur nature apporterait de nombreuses informations sur le passage à l'échelle de la méthode et permettrait de modifier, si besoin, la manière de traiter les données. Techniquement, il conviendrait de développer des interfaces de programmes assistant l'utilisation dans sa complétion de schéma ou dans sa complétion de données par navigation sur la famille de treillis.

D'autres codages des données pourraient être testés (n'utiliser que l'information de destination des rôles par exemple), ou diviser les rôles entre deux mêmes ensembles de descripteurs selon des catégories.

Un travail reste à mener pour l'analyse préliminaire des informations de Dbpedia pour choisir les catégories initiales et le codage des propriétés en attributs ou en rôles. Nous avons mentionné que les attributs des descripteurs créés pourraient être réifiés afin d'être eux-mêmes classés et que cela permettrait d'enrichir le regroupement des descripteurs. Ces attributs pourraient être décrits par leurs noms, leur type, etc.

Comme les regroupements dépendent fortement des termes tels que les noms des propriétés par exemple, une utilisation de connaissances lexicales comme la synonymie, l'hyperonymie ou d'autres semble indispensable pour avoir des résultats plus pertinents.

ANNEXE 1 EXTRAITS D'INSTANCE D'ÉCRIVAINS, LIVRES ET LIEUX

Pour chaque type d'entités, une sélection de propriétés a été effectuée. Des « ... » ont été placés pour éviter de recopier toutes les valeurs qui disparaîtront à l'étape d'extraction des descripteurs.

- Entités de type Person (http://dbpedia.org/ontology/Person):

Edgar Poe (http://dbpedia.org/page/Edgar_Allan_Poe)
dbpprop:name Edgar Allan Poe
dbpedia-owl:birthDate 1809-01-19
dbpedia-owl:birthPlace **dbpedia:Boston**
dbpedia-owl:deathDate 1849-10-07
dbpedia-owl:deathPlace **dbpedia:Baltimore**

Mary Shelley (http://fr.dbpedia.org/page/Mary_Shelley)
dbpprop:name Mary Shelley
dbpedia-owl:birthDate 1797-08-30
dbpedia-owl:birthPlace **dbpedia:London**
dbpedia-owl:deathDate 1851-02-01
dbpedia-owl:deathPlace **dbpedia:Bournemouth**

Stephen King (http://dbpedia.org/page/Stephen_King)
dbpprop:name Stephen King
dbpedia-owl:activeYearsStartYear 1967-01-01
dbpedia-owl:birthDate 1947-09-21
dbpedia-owl:birthPlace **dbpedia:Portland,_Maine**
dbpedia-owl:genre dbpedia:Horror_fiction
dbpprop:awards dbpedia:National_Book_Award dbpedia:World_Fantasy_Award
dbpprop:pseudonym Richard Bachman, John Swithen
dbpprop:website http://www.stephenking.com

Brian Keene (http://dbpedia.org/page/Brian_Keene)
dbpprop:name Brian Keene
dbpedia-owl:activeYearsStartYear 1997-01-01
dbpedia-owl:birthDate 1967-01-01
dbpedia-owl:birthPlace **dbpedia:Pennsylvania**
dbpedia-owl:genre dbpedia:Horror_fiction
dbpprop:website http://www.briankeene.com/
dbpedia-owl:nationality dbpedia:United_States

- Entités de type book (http://dbpedia.org/ontology/Book):

Frankenstein (http://dbpedia.org/page/Frankenstein)
dbpedia-owl:abstract text
dbpedia-owl:author **dbpedia:Mary_Shelley**
dbpprop:language English
dbpprop:genre dbpedia:Gothic_fiction

The Narrative of Arthur Gordon Pym of Nantucket
(http://dbpedia.org/page/The_Narrative_of_Arthur_Gordon_Pym_of_Nantucket)
dbpedia-owl:abstract text
dbpedia-owl:author **dbpedia:Edgar_Allan_Poe**
dbpprop:language English
dbpprop:publisher dbpedia:Harper_(publisher)

Six stories (http://dbpedia.org/page/Six_Stories)
dbpedia-owl:abstract text
dbpedia-owl:author **dbpedia:Stephen_King**
dbpprop:language dbpedia:English_language
dbpprop:publisher dbpedia:Philtrum_Press
dbpedia-owl:mediaType dbpedia:Paperback
dbpedia-owl:wikiPageExternalLink http://www.horrorking.com/sixstory.html
dbpprop:coverArtist Michael Alpert

dbpedia-owl:literaryGenre dbpedia:Horror_fiction

Dead Sea (novel) (http://dbpedia.org/page/Dead_Sea_(novel))
dbpedia-owl:abstract text
dbpedia-owl:author **dbpedia:Brian_Keene**
dbpprop:language dbpedia:English_language
dbpprop:publisher dbpedia:Delirium_Books
dbpedia-owl:mediaType dbpedia:Hardcover
dbpedia-owl:literaryGenre dbpedia:Horror_fiction
dbpprop:hasPhotoCollection http://wifo5-03.informatik.uni-mannheim.de/flickrwrappr/photos/Dead_Sea_(novel)

- Entités de type place (http://dbpedia.org/ontology/Place)

England (http://dbpedia.org/page/England)
dbpedia-owl:areaTotal ....
dbpedia-owl:leaderTitle ....
dbpedia-owl:flag ....

United States (http://dbpedia.org/page/United_States)
dbpedia-owl:areaTotal 9826675000000.000000 (xsd:double)
dbpedia-owl:leaderTitle President
dbpedia-owl:flag Flag of the United States.svg

Washington,_D.C. (http://dbpedia.org/page/Washington,_D.C.)
dbpedia-owl:areaTotal ....
dbpedia-owl:area 1.000000 (xsd:double)
dbpedia-owl:leaderTitle ....
dbpedia-owl:flag ....
dbpedia-owl:postalCode 20001-20098, 20201-20599

Boston (http://dbpedia.org/page/Boston)
dbpedia-owl:areaTotal ....
dbpedia-owl:area ....
dbpedia-owl:leaderTitle ....
dbpedia-owl:flag ....
dbpedia-owl:postalCode .....

Baltimore (http://dbpedia.org/page/Baltimore)
dbpedia-owl:areaTotal ....
dbpedia-owl:area ....
dbpedia-owl:leaderTitle ....
dbpedia-owl:flag ....
dbpedia-owl:postalCode .....

London (http://dbpedia.org/page/London)
dbpedia-owl:areaTotal ....
dbpedia-owl:area ....
dbpedia-owl:leaderTitle ....
dbpedia-owl:flag ....
dbpedia-owl:postalCode .....

Bournemouth (http://dbpedia.org/page/Bournemouth)
dbpedia-owl:areaTotal ....
dbpedia-owl:area ....
dbpedia-owl:leaderTitle ....
dbpedia-owl:flag ....
dbpedia-owl:postalCode .....

*Portland,_Maine (*http://fr.dbpedia.org/page/Portland_(Maine))
dbpedia-owl:areaTotal ....
dbpedia-owl:area ....
dbpedia-owl:leaderTitle ....

dbpedia-owl:flag ....

dbpedia-owl:postalCode .....

Pennsylvania (http://dbpedia.org/page/Pennsylvania)

dbpedia-owl:areaTotal ... (xsd:double)

dbpedia-owl:leaderTitle Mayor

dbpedia-owl:flag ....

ANNEXE 2 DESCRIPTEURS RETENUS POUR NOTRE EXEMPLE

- Entités de type Person (http://dbpedia.org/ontology/Person):

Edgar Poe (http://dbpedia.org/page/Edgar_Allan_Poe)
dbpprop:name
dbpedia-owl:birthDate
dbpedia-owl:birthPlace **dbpedia:Boston**
dbpedia-owl:deathDate
dbpedia-owl:deathPlace **dbpedia:Baltimore**

Mary Shelley (http://fr.dbpedia.org/page/Mary_Shelley)
dbpprop:name
dbpedia-owl:birthDate
dbpedia-owl:birthPlace **dbpedia:London**
dbpedia-owl:deathDate
dbpedia-owl:deathPlace **dbpedia:Bournemouth**

Stephen King (http://dbpedia.org/page/Stephen_King)
dbpprop:name
dbpedia-owl:activeYearsStartYear
dbpedia-owl:birthDate
dbpedia-owl:birthPlace **dbpedia:Portland,_Maine**
dbpedia-owl:genre
dbpprop:awards
dbpprop:pseudonym
dbpprop:website

Brian Keene (http://dbpedia.org/page/Brian_Keene)

dbpprop:name

dbpedia-owl:activeYearsStartYear

dbpedia-owl:birthDate

dbpedia-owl:birthPlace **dbpedia:Pennsylvania**

dbpedia-owl:genre

dbpprop:website

dbpedia-owl:nationality

- Entités de type book (http://dbpedia.org/ontology/Book):

Frankenstein (http://dbpedia.org/page/Frankenstein)

dbpedia-owl:abstract

dbpedia-owl:author **dbpedia:Mary_Shelley**

dbpprop:language

dbpprop:genre

The Narrative of Arthur Gordon Pym of Nantucket
(http://dbpedia.org/page/The_Narrative_of_Arthur_Gordon_Pym_of_Nantucket)

dbpedia-owl:abstract

dbpedia-owl:author **dbpedia:Edgar_Allan_Poe**

dbpprop:language

dbpprop:publisher

Six stories (http://dbpedia.org/page/Six_Stories)

dbpedia-owl:abstract

dbpedia-owl:author **dbpedia:Stephen_King**

dbpprop:language

dbpprop:publisher

dbpedia-owl:mediaType

dbpedia-owl:wikiPageExternalLink

dbpedia-owl:literaryGenre

Dead Sea (novel) (http://dbpedia.org/page/Dead_Sea_(novel))
dbpedia-owl:abstract
dbpedia-owl:author **dbpedia:Brian_Keene**
dbpprop:language
dbpprop:publisher
dbpedia-owl:mediaType
dbpedia-owl:literaryGenre
dbpprop:hasPhotoCollection

- Entités de type place (http://dbpedia.org/ontology/Place)

England (http://dbpedia.org/page/England)
dbpedia-owl:areaTotal
dbpedia-owl:leaderTitle
dbpedia-owl:flag

United States (http://dbpedia.org/page/United_States)
dbpedia-owl:areaTotal
dbpedia-owl:leaderTitle
dbpedia-owl:flag

Washington,_D.C. (http://dbpedia.org/page/Washington,_D.C.)
dbpedia-owl:areaTotal
dbpedia-owl:area
dbpedia-owl:leaderTitle
dbpedia-owl:flag
dbpedia-owl:postalCode

Boston (http://dbpedia.org/page/Boston)
dbpedia-owl:areaTotal
dbpedia-owl:area

dbpedia-owl:leaderTitle
dbpedia-owl:flag
dbpedia-owl:postalCode

Baltimore (http://dbpedia.org/page/Baltimore)
dbpedia-owl:areaTotal
dbpedia-owl:area
dbpedia-owl:leaderTitle
dbpedia-owl:flag
dbpedia-owl:postalCode

London (http://dbpedia.org/page/London)
dbpedia-owl:areaTotal
dbpedia-owl:area
dbpedia-owl:leaderTitle
dbpedia-owl:flag
dbpedia-owl:postalCode

Bournemouth (http://dbpedia.org/page/Bournemouth)
dbpedia-owl:areaTotal
dbpedia-owl:area
dbpedia-owl:leaderTitle
dbpedia-owl:flag
dbpedia-owl:postalCode

*Portland,_Maine (*http://fr.dbpedia.org/page/Portland_(Maine))
dbpedia-owl:areaTotal
dbpedia-owl:area
dbpedia-owl:leaderTitle
dbpedia-owl:flag
dbpedia-owl:postalCode

Boulogne-Billancourt (http://dbpedia.org/page/Boulogne-Billancourt)

dbpedia-owl:areaTotal

dbpedia-owl:area

dbpedia-owl:leaderTitle

dbpedia-owl:flag

dbpedia-owl:postalCode

Pennsylvania (http://dbpedia.org/page/Pennsylvania)

dbpedia-owl:areaTotal

dbpedia-owl:leaderTitle

dbpedia-owl:flag

ANNEXE 3 FAMILLE DE CONTEXTES CONSTRUITE POUR L'EXEMPLE ILLUSTRATIF

Dans cette famille de contextes, l'opérateur de scaling existentiel est utilisé pour les contextes relationnels et à toutes les étapes.

Contexte formel Writer

	name	birthDate	deathDate	activeYearsStartYear	genre	awards	pseudonym	website	nationality
EdgarPoe	X	X	X						
MaryShelley	X	X	X						
StephenKin	X	X		X	X	X	X	X	X
BrianKeene	X	X		X	X			X	X

Contexte formel Book

	abstract	language	genre	publisher	mediaType	wikiExtLink	coverArtist	literaryGenre	hasPhotoCollection
Frankenstein	X	X	X						
ArthurGordonPym	X	X		X					
SixStories	X	X		X	X	X	X	X	
DeadSea	X	X		X	X	X		X	X

Contexte formel Place

	areaTotal	leaderTitle	flag	area	postalCode
England	X	X	X		
UnitedStates	X	X	X		
WashingtonDC	X	X	X	X	X
Boston	X	X	X	X	X
Baltimore	X	X	X	X	X
London	X	X	X	X	X
Bournemouth	X	X	X	X	X
PortlandMaine	X	X	X	X	X
Pennsylvania	X	X	X		

Contexte formel RoleWriterPlace

	birthPlace	deathPlace	place
EdgarPoe::birthPlace	X		X
MaryShelley::birthPlace	X		X
StephenKing::birthPlace	X		X
BrianKeene::birthPlace	X		X
EdgarPoe::deathPlace		X	X
MaryShelley::deathPlace		X	X

Contexte formel RoleBookWriter

	author
Frankenstein::author	X
ArthurGordonPym::author	X
SixStories::author	X
DeadSea::author	X

Contexte relationnel hasRoleWriterPlace
Source Writer
Target RoleWriterPlace
Scaling exist

	EdgarPoe::birthPlace	MaryShelley::birthPlace	StephenKing::birthPlace	BrianKeene::birthPlace	EdgarPoe::deathPlace	MaryShelley::deathPlace
EdgarPoe	X				X	
MaryShelley		X				X
StephenKing			X			
BrianKeene				X		

Contexte relationnel hasOriginWriterPlace
Source RoleWriterPlace
Target Writer
Scaling exist

	EdgarPoe	MaryShelley	StephenKing	BrianKeene
EdgarPoe::birthPlace	X			
MaryShelley::birthPlace		X		
StephenKing::birthPlace			X	
BrianKeene::birthPlace				X
EdgarPoe::deathPlace	X			
MaryShelley::deathPlace		X		

Contexte relationnele hasDestinationWriterPlace
Source RoleWriterPlace
Target Place
Scaling exist

	England	UnitedStates	WashingtonDC	Boston	Baltimore	London	Bournemouth	PortlandMaine	Pennsylvania
EdgarPoe::birthPlace				X					
MaryShelley::birthPlace						X			
StephenKing::birthPlace								X	
BrianKeene::birthPlace									X
EdgarPoe::deathPlace					X				
MaryShelley::deathPlace							X		

Contexte relationnel hasRoleBookWriter
Source Book
Target RoleBookWriter
Scaling exist

	Frankenstein::author	ArthurGordonPym::author	SixStories::author	DeadSea::author
Frankenstein	X			
ArthurGordonPym		X		
SixStories			X	
DeadSea				X

Contexte relationnel hasOriginBookWriter
Source RoleBookWriter
Target Book
Scaling exist

	Frankenstein	ArthurGordonPym	SixStories	DeadSea
Frankenstein::author	X			
ArthurGordonPym::author		X		
SixStories::author			X	
DeadSea::author				X

Contexte relationnel hasDestinationBookWriter
Source RoleBookWriter
Target Writer
Scaling exist

	EdgarPoe	MaryShelley	StephenKing	BrianKeene
Frankenstein::author		X		
ArthurGordonPym::author	X			
SixStories::author			X	
DeadSea::author				X

BIBLIOGRAPHIE

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. et Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. *The Semantic Web*. 4825:722-735.

Baader, F., Calvanese, D., McGuinness, L., Nardi, D. et Patel-Schneider, P.F. (2003). *The description logic handbook: theory, implementation, and applications*. : Cambridge university press.

Bernhard, G. et Rudolf, W. (1999). *Formal Concept Analysis: Mathematical Foundations*. Springer Science & Business Media.

Candan, K.S., Liu, H. et Suvarna, R. (2001). Resource description framework: metadata and its applications. *ACM SIGKDD Explorations Newsletter, 3*(1):6-19.

Chekol, M. W. et Napoli, A. (2013). An FCA framework for knowledge discovery in SPARQL query answers. *The 12th International Semantic Web Conference*, 1035:197-200.

Curé, O., Lamolle, M. et Duc, C.L. (2013). Ontology based data integration over document and column family oriented nosql. *arXiv preprint arXiv:1307.2603*.

d'Aquin, M. et Motta, E .(2011). Extracting relevant questions to an RDF dataset using formal concept analysis. *Proceedings of the sixth international conference on Knowledge capture*, pp. 121-128.

Dau, F. et Klinger, J. (2005). From formal concept analysis to contextual logic. *Formal Concept Analysis*, LNAI 3626:81-100.

David, L. et Scharffe, F. (2012). Détection de clefs pour l'interconnexion et le nettoyage de jeux de données. *Ingénierie des Connaissances*, pp. 401-415

Delteil, A., Faron, C. et Dieng, R. (2002). Building concept lattices by learning concepts from rdf graphs annotating web documents. *Conceptual Structures: Integration and Interfaces*, pp. 191-204.

Fennouh, S., Nkambou, R., Valtchev, P. et Rouane-Hacene, M. (2015). On the Assessment of Concept Relevance in FCA-based Ontology Restructuring. A parraitre dans *International Conference on Tools with Artificial Intelligence (ICTAI)*.

Hacene, M.R. (2007). *Etude de l'analyse formelle dans les donnees relationnelles. Application a la restructuration des modeles structuraux UML*. ProQuest.

Hacene, M.R., Napoli, A., Valtchev, P., Toussaint, Y. et Bendaoud, R. (2008). Ontology learning from text using relational concept analysis. *e-Technologies, 2008 International MCETECH Conference on*, pp. 154-163.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P. et Auer, S. (2014). DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, *5*:1-29.

Godin, R. et Missaoui, R. (1994). An incremental concept formation approach for learning from databases. *Theoretical computer science*, *133*(2):387-419.

McBride, B. (2001). *Jena: Implementing the RDF Model and Syntax Specification*. SemWeb, pp. 74-83.

Mehri-Dehnavi, R. (2014). *Inferring missing schema from linked data using formal concept analysis (FCA)*. Maîtrise, Université Du Québec à Montréal (UQAM).

Ganter, B. et Wille, R. (1999). *Formal Concept Analysis*. Springer, mathematical foundations edition.

Godin, R. (2006). *Systèmes de gestion de bases de données par l'exemple (éd. 2)*. Longueuil, Québec, Canda: Loze-Dion.

Miller, E. (1998). An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, *25*(1):15-19.

Miralles, A., Dolques, X., Huchard, M., Le Ber, F., Libourel, T., Nebut, C. et Osman-Guédi, A. (2014). Exploration de la factorisation d'un modèle de classes sous contrôle des acteurs. *INFORSID*, pp. 245-261.

Morsey, M., Lehmann, J., Auer, S., Stadler, C. et Hellmann, S. (2012). Dbpedia and the live extraction of structured data from wikipedia. *Program, 46*(2):157-181.

Nikolov, A., Uren, V. et Motta, E.(2007). KnoFuss: A comprehensive architecture for knowledge fusion. *Proceedings of the 4th international conference on Knowledge capture*, pp. 185-186.

Nikolov, A., Uren, V. et Motta, E. (2010). Data linking: Capturing and utilising implicit schema-level relations. *Workshop on Linked Data On the Web (LDOW)*.

Pan, J.Z. (2009). Resource description framework. *Handbook on Ontologies*, pp. 71-90.

Paulheim, H. et Fümkranz, J. (2012). Unsupervised generation of data mining features from linked open data. *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, pp. 31-42.

Rouane-Hacene, M., Huchard, M., Napoli, A. et Valtchev, P. (2013). Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 67(1):81-108.

Salvadores, M., Correndo, G., Rodriguez-Castro, B., Gibbins, N., Darlington, J. et Shadbolt, N.R. (2009). Linksb2n: Automatic data integration for the semantic web. *On the Move to Meaningful Internet Systems: OTM 2009*, LNCS 5371:1121-1138.

Scharffe, F., Liu, Y. et Zhou, C. (2009). Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR)*.

Seligman, L. et Roenthal, A. (2001). XML's impact an databases and data sharing. *Computer*, 34(6):59-67.

Sertkaya, B. (2011). A survey on how description logic ontologies benefit from formal concept analysis. *arXiv preprint arXiv:1107.2822*.

Stumme, G. et Maedche, A.. (2001). Ontology merging for federated ontologies on the semantic web. *Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001)*, pp. 413-418.

Völker, J. et Niepert, M. (2011). Statistical schema induction. T*he Semantic Web: Research and Applications*, LNCS 6643:124-138.

Wienand, D. et Paulheim, H. (2014). Detecting incorrect numerical data in dbpedia. *The Semantic Web: Trends and Challenges*, LNCS 8455:504-518.

Wille, R. (2009). Restructuring lattice theory: an approach based on hierarchies of concepts. *ICFCA*, LNAI 5548:314-339.